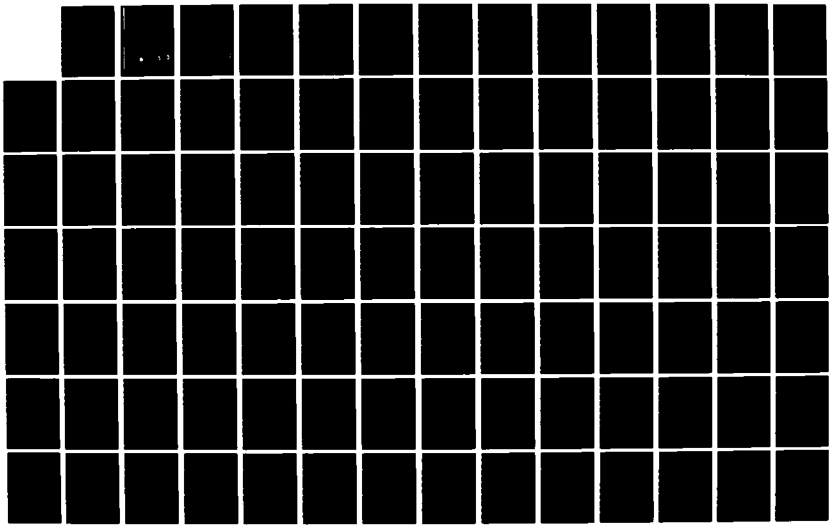
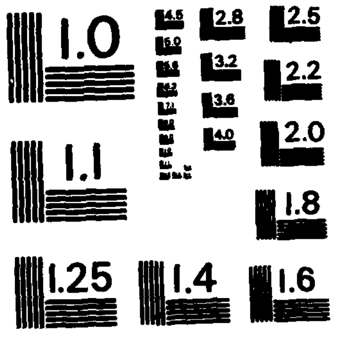
AD-A159 980 PROJECT SCHEDULING IN PROJECT-ORIENTED PRODUCTION SYSTEMS(U) CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER A DINCERLER SEP 85 ORC-85-11 1/2

UNCLASSIFIED F/G 13/8 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

12

OPERATIONS RESEARCH CENTER

PROJECT SCHEDULING
IN PROJECT-ORIENTED PRODUCTION SYSTEMS

by

Abdurrezak Dincerler*

ORC 85-11 September 1985

UNIVERSITY OF CALIFORNIA

BERKELEY

PROJECT SCHEDULING IN PROJECT-ORIENTED PRODUCTION SYSTEMS

by

Abdurrezak Dincerler*

ORC-85-11 ✓ September 1985

*Operations Research Center, University of California, Berkeley, California.

This research was supported by the Office of Naval Research under Contract N00014-76-C-0134 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ORC 85-11	2. GOVT ACCESSION NO. AD-A159980	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROJECT SCHEDULING IN PROJECT-ORIENTED PRODUCTION SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Abdurrezak Dincerler		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0134
9. PERFORMING ORGANIZATION NAME AND ADDRESS Operations Research Center University of California Berkeley, CA 94720		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 337 015
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Dept. of the Navy Arlington, Virginia 22217		12. REPORT DATE September 1985
		13. NUMBER OF PAGES 130
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Project Management	Rework
Production Scheduling	Simulation
Critical Path Method	Dynamic Activity Analysis
Resource Constraints	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(See Abstract)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PROJECT SCHEDULING IN PROJECT-ORIENTED PRODUCTION SYSTEMS

Abdurrezak Dincerler

D.Eng. Industrial Engineering and Operations Research

Sponsor: Office of Naval Research

Robert C. Leachman
Chairman of Committee

Abstract

This thesis details a new method of scheduling project activities in project-oriented production systems. The scheduling problem in a project-oriented production system can be characterized as a part of the hierarchical planning and scheduling process in which an aggregate planning level allocates resources among projects of the production system and specifies major due dates of project executions. The goal of project scheduling is to minimize the project completion time subject to resource capacities and due dates which are specified by the aggregate planning level. Resource capacities for a particular project are time-varying but inflexible at any particular time. The method of scheduling aims to reflect the characteristics of work in ship overhauls in a shipyard which is believed to be a typical example of a project-oriented production system. The method is designed to be computationally feasible and managerially practical.

Although there are various methods and techniques for project scheduling which have been proposed in the literature, none fulfills all the needs posed by the shipyard's scheduling problem. Additional keywords: algorithms; personnel management; Ship overhauling, [illegible]

An important characteristic of work in a shipyard is that activities of projects may have variable rates of resource applications. The actual duration of an activity depends on the rate of resource applications. A model with variable *intensities* for activities is introduced to cope with this work characteristic. Using the variable intensity model, heuristic intensity assignment algorithms are developed for two different management policies concerning the flexibility of resource redeployment. The intensity assignment algorithms work as decision makers which decide activity start times and intensity levels, ie, resource allocations, for activities. The validity of the algorithms is established by comparing their performances to the performances of two methods adapted from the methods in literature in comparative experiments with arbitrarily generated project parameters.

The other important characteristic of work in ship overhauling is the uncertainty of work content of the activities and also the uncertainty in the project network due to the prospect of repair rework which may be needed after testing of subsystems. To handle these uncertainties, a simulation model which reflects the stochastic nature of work and uses the intensity assignment algorithms as a decision maker is introduced.

Finally, the intensity assignment algorithms and the simulation model are used to generate several schedules for an actual large ship overhaul project as an implementation of the method proposed. Results are discussed.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Professor Robert C. Leachman for his guidance, encouragement and constant inspiration throughout the progress of this research. I am grateful to Professor Roger Glassey and Professor Kieth Crandall who served in my thesis committee. The research reported in this thesis was supported by the Office of Naval Research and Puget Sound Naval Shipyard under contract with the University of California-Berkeley.

TABLE OF CONTENTS

1. INTRODUCTION

Scheduling large projects is a very important task in project-oriented production systems, such as shipyards and airplane construction firms. We will refer to a production system that completes a number of different types of large projects on a repetitive basis as their outputs as a project-oriented production system. Such systems are characterized by inflexible capacities for scarce resources, particularly various types of skilled trade labor. These resources must be utilized efficiently in multiple, concurrent projects in order to maximize throughput of the system.

The scheduling of individual projects has been given considerable attention from both theoretical and practical people because of its substantial financial impact. In general, the cost of large projects is very high and even a small reduction in project time may realize substantial savings for a company.

A number of methods have been developed and a large amount of computer software has been marketed in the last three decades for the project scheduling problem. In general, methods developed are tailored to schedule the projects for construction firms where the resource capacities are flexible and the projects undertaken by those companies are not similar to one another.

The scheduling problem in a project-oriented production system can be characterized as a part of the hierarchical planning and scheduling process in which an aggregate planning level allocates resources among projects of the production system and specifies major due dates of project executions. The goal of project scheduling is to minimize the project completion time subject to the resource allocations and due dates which are specified by the aggregate planning level. The resource allocations to a particular project serve as resource capacities for the project. These capacities are time-varying but inflexible in a particular time.

The lack of a scheduling method that captures the needs of project-oriented production systems led to the development of the method described herein for scheduling projects in such systems. Chapter 2 describes the scheduling environment in project-oriented production systems. Differences between these systems and one-time project systems are identified.

A review of project scheduling methods is given in Chapter 3. The review reflects the historical development of the project scheduling problem. Resource Unconstrained Project Scheduling, Resource Constrained Project Scheduling, and Generalized Project Scheduling methods are reviewed.

Modeling assumptions for a new planning and scheduling system in a project-oriented production environment are given in Chapter 4. A generalized model for Resource Constrained Project Scheduling is introduced in Chapter 5. In this model, variable resource application levels which we call variable *intensities* for each activity are used. A linear activity analysis model is used to characterize variable intensities for activities. Intermediate product transfers between activities are analyzed.

Chapter 6 provides the development of intensity assignment algorithms for two different management policies. The intensity assignment algorithms work as decision-makers which decide activity start times and intensity levels. In the first policy, management does not allow reduction in the level of resources which is committed to an activity once it is started, although it allows an increase in this level while the activity is in progress (Upgrading-Only Algorithm). In the second policy, management allows reductions in the level of resources that is committed to an activity as well as increases in it (Upgrading-Downgrading Algorithm). Both algorithms use the same procedure to handle time varying resource capacities which is explained in section 6.5.

Chapter 7 demonstrates the validity of the intensity assignment algorithms. Two other scheduling methods from the literature and the intensity assignment algorithms are all used to schedule arbitrarily generated projects. (The two methods used in comparison are described.) The procedure for arbitrary network generation is summarized. Comparative results of experiments are analyzed.

Chapter 8 proposes new methods for handling stochastic elements in project scheduling, focusing on stochastic resource requirements and network structure. Stochastic parameters are described. Treatments for these parameters during simulations are explained.

Finally, intensity assignment algorithms are utilized to simulate a ship overhaul project and results are presented. The implications of the results for shipyard management are given in Chapter 9.

2. SCHEDULING IN PROJECT-ORIENTED PRODUCTION SYSTEMS

Project-oriented production systems produce large similar projects as its products. In order to accomplish the completion of one unit of such a product, a large number of interrelated activities are performed in sequence. Each activity requires work from various trade labor types. As an example of such a production system, a shipyard for overhauling naval ships can be given. At any given time ten to fifteen ships may be in various of stages of overhaul in such a shipyard. Overhaul of different ships in the same class have very similar tasks. Overhaul of a ship may consist of thousands of activities. Each activity may require work effort from hundreds of different trade labor shops.

2.1. Production System versus One-time Projects

We view a production system as a system which uses a stream of inputs (labor of different types , raw material, money, production facilities including equipment and machinery) in order to produce some design outputs (projects in our case). In a production system there are permanent facilities and permanent resources (trade labor) in the system. Those resources are utilized by the projects which are in progress. Note that, even if there is no production, the cost of those resources and facilities must be absorbed by the system.

The project cost for the production system largely depends on project time and allocation of the system costs among projects. The objective of planning and scheduling the projects and project activities in such a system is to take on and finish the projects as early as possible within the given resource capacities. Equivalently, the objective is to maximize the throughput, ie, the number of projects completed by the system within a given (rolling) time horizon.

We will refer to projects that are executed once (or perhaps a small number of times) as one-time projects. In systems performing one-time projects the facilities and resources which will be used in a project are obtained temporarily. The acquisition or rental of facilities and resources are made after a decision is made to take on the project. Timing and amounts of resources that will be acquired for the project can be planned in advance. Also, supplementary resources may be acquired during the project execution if any bottleneck may arise. Most of the construction projects (building bridges, dams, etc.) can be considered as one-time projects since construction companies do not hold substantial amounts of permanent resources. For one-time projects, project cost largely depends on the cost of the resources that are temporarily acquired. The objective of planning and scheduling of one-time projects is to minimize the cost of the project while keeping the promised project due dates.

The research reported here is the development of some techniques to assist project scheduling in project-oriented production systems. The research is motivated by a study of ship overhaul scheduling within a naval shipyard.

2.2. Single Project versus Multi-Project Environments

In a single project environment, the production system executes only one project at a time. Total resources of the production system are available to activities of the project which is in execution.

In a multi-project environment, there are simultaneous or overlapping projects which are competing for the same resources. For scheduling purposes, one might consider treating the collection of projects as one big project. This can be practical if the projects are not too big; otherwise the scheduling problem becomes intractable.

Boysen (1982) and Leachman and Boysen (1984) have proposed management in large multi-project environments using multiple levels of planning and scheduling effort. Planning and scheduling in a multi-project production environment can be viewed from the perspective of both project management and resource management. Effective planning and scheduling will only be achieved if those two perspectives are properly linked together.

Planning projects and resources is a complex task in multi-project production environments, in which hierarchical levels of planning must necessarily exist. Boysen (1982) proposes three levels of planning in the planning process in such a system.

These levels are as follows:

1. Inter-Project Planning
2. Individual Project Planning
3. Project Scheduling

Going from higher levels to lower levels, additional constraints are imposed, the planning horizon becomes shorter, there is more specific information, and the degree of uncertainity decreases. Explanations of decisions involved in each planning level are provided in section 2.3.

2.3. Project Planning

Inter-Project Planning

Long range planning is needed to fit new projects into the system. The timing and compression of the projects largely depends on the resource loadings of each project. A new project "fits in" if a consistent and feasible plan for manning the new project and all other projects is developed. Expected due dates of some important milestones of the project must be estimated. On the basis of the inter-project plan, time-dependent project progress workload forecasts are given to resource managers.

Individual Project Planning

The aim of this level is to track and control project execution to keep projects on schedule. Projects are described in more detail in this level. Certain project due dates will have been established at the multi project planning level. Additional milestones are then defined to indicate start and finish of important phases of the project.

A detailed network of activities is developed to observe the progress between milestones more accurately. Duration and resource requirements for each activity are estimated. Using these estimates for durations, predictions of milestone actualization dates can be made more accurately. Flexibility in performing the project activities can be determined.

Total resource requirements of the project activities over the project duration are forecasted more accurately. The feasibility of the allocation of resources to the project in the inter-project analysis may be verified and/or modifications of input parameters for inter-project analysis may be made. Using detailed activity networks it is easier and more accurate to track and control the project.

Project Scheduling

This level involves day-to-day operating decisions. The detailed activity network is used to establish the priorities among the project activities. The resources preallocated to the project by higher-level planning are allocated among the activities of the project using the scheduling priorities. Scheduling of activities is constrained by their precedence relations, preassigned due dates and preallocation of resources of each type. A daily schedule shows the activities that will be in execution and the amounts of resources used by them.

The project's execution status is monitored and updated. The status can be measured in terms of progress in activities. The initial durations and resource requirements of activities are revised if necessary. Delay in activities is identified.

The research reported here deals with the project scheduling level. Improvement in the scheduling level will facilitate improvement of the planning process in all levels. It serves to complement the work of Boysen (1982) and Leachman and Boysen (1984) on the methodology for inter-project planning.

2.4. Example of Industry Practice: Shipyard's Approach

A shipyard for overhauling naval ships is examined as an example of a project-oriented production system.

There are three hierarchical planning levels for project planning and scheduling in the shipyard.

1. Inter-Project Planning
2. Project Planning
3. Project Control

The inter-project planning level decides the timing of future ship overhauls for a horizon of a few years. The decision process can be described as follows.

The shipyard's planning staff has rough information of work requirements of future overhauls. The source of the information is primarily past experience, since ships overhauled belong to specific classes and ships in the same class are very similar. Each ship overhaul has several alternative loading curves for total manpower. The shapes of those curves are examples from past history (when available), scaled to the requirements of the upcoming overhaul. The staff initially selects a loading curve for each future overhaul. Using these curves, a rough-cut capacity analysis is performed to establish "reasonable" start and due dates for future overhauls. The loading curves (ie., load histories of the sum of all trade labor shops) for ongoing and future overhauls are juxtaposed and tabulated according to the trial start and finish dates for each overhaul. The resulting total load is compared to total shipyard manpower. Ideally, the selection of loading curves would be made so that the sum of manpower load across overhauls fits the "capacity" of the yard within "reasonable" tolerance. Where major discrepancies exist, curves are shifted or substituted until one comes up with "reasonable" start and due dates for future projects.

Once these rough start and finish times of projects are decided by the planning staff, the workload forecasting department prepares "Workload Forecasts" corresponding to the total manpower curve selected for each overhaul. Summing up each resource across overhauls according to the selected resource loading patterns gives the workload forecast for that resource for that planning horizon.

Forecasts of total resource workload and the resource workload of each overhaul are given to resource managers so that they can make adjustments of their work-force. Ordinarily, no major adjustments in work-force levels are routinely allowed except the timing of labor vacations.

As the starting time for an upcoming overhaul draws nearer, project planning is undertaken, whereby the work requirements are defined and estimated in more detail. A critical path network is developed and activity durations are estimated. Due dates for major project events are established by judgement of senior management.

The Project Control level aims to track and control the progress of overhauls in order to finish the overhauls at planned due dates, and within resource capacities.

Each overhaul is assigned a project manager who is responsible to track and control the project. The scheduling department helps the project manager to control the project. It prepares a detailed activity network for the project. It does ordinary critical path method calculations for the overhaul by using the due dates for major events specified in project planning as constraints to schedule the activities. Schedules developed here are not necessarily consistent with resource capacities, since CPM calculations are made assuming that there are enough resources in the system. The scheduling department updates the CPM calculations periodically.

For each activity, the shop with the most work required is designated as the *lead shop* for the activity. Each trade labor shop has a manager who assigns labor among activities of projects. Every shop manager is furnished with a list of priorities among the activities on the basis of least total float of activities from CPM calculations. The shop manager allocates manpower from all shops among available activities for which his shop is designated as lead shop. In general, activities can be manned in many different ways. The lead shop managers are the ones who decide the way of manning the activities.

The project manager is responsible for controlling the project execution. If any critical activity is delayed because a resource needed by the activity is not allocated by a shop manager, the project manager reports it to top management. Top management may resolve the conflict between project managers and shop managers by establishing priorities between projects which are competing for the resource. It may overrule the allocations made by the shop manager in the case that the project manager successfully appeals to top management.

2.5. Special Characteristics of Work

Flexible Loading of Resources

Many activities of projects in the shipyard may be manned in many different ways. Lead shop managers have a large amount of flexibility in manning activities. The total resources required by the activity must be satisfied but the rate and pattern of resource application is decided by the shop manager. He has the flexibility of allocating as many men as is practical if he thinks the job is urgent or he may allocate a smaller crew otherwise. The actual duration of the job depends on the rate of resource applications.

Uncertain Work Content

Each activity of a project has authorized work from the shops whose services are required. The authorized work required by an activity from the shops is defined in terms of the tasks to be accomplished, but in actuality there is considerable uncertainty in the total amount of resources required, ie, the total man-hours to accomplish the tasks required by the job. Historical data on total man-hours from various shops required by each activity is available from which distributions for total man hours can be developed.

Besides authorized work from the shops for the project activities, the need for rework or previously unauthorized work may be discovered during testing of the subsystems that have been worked on before. Testing of the subsystems is a substantial part of the overhauling project. In the shipyards rework of authorized work or newly-discovered (unauthorized) work occurs frequently.

Needs of Shipyard Management

Shipyard management needs a system design for project planning and scheduling which reflects the characteristics of work in overhauling which are discussed above and which fits in with inter-project planning. Proposed methods should be computationally feasible and managerially practical. The study reported here is motivated by the needs of shipyard management. The discussion of the proposed system is deferred until Chapter 4 in order to review previously-developed scheduling methods (Chapter 3).

3. REVIEW OF PROJECT SCHEDULING PROBLEMS AND SOLUTION METHODS

3.1. Overview

In general, a large complex project involves a large number of component activities that are dependent on each other by specified precedence requirements. Scheduling the component activities in order to achieve a certain goal is the main task of project management. The goal of scheduling may be, completing the project by a specific deadline, or minimizing the cost of the project in order to meet the specified target due date, or minimizing the project completion time, etc. The constraints imposed on scheduling of a project may also vary depending on the specific requirements of the project.

Methods of scheduling may vary according to their specific goal and constraints imposed on them but they have an important feature in common: all of them are based on the representation of project as a network of activities. Therefore, they often referred to as *network analysis*, *network planning*, and *network scheduling*. The most widely used names are CPM and PERT, which stand for two techniques that evolved in the late fifties, and they will be covered in section 3.2.

Activity Networks

A project can be represented by an activity network in which nodes represent activities and arcs represent precedence relations between activities. This representation is called as Activity-on-Node. Activity-on-Arc representation is also used but it requires introduction of dummy activities to show some precedence relations which is not needed A-O-N representations†.

Projects are decomposed into activities using several criteria. Resource use, work location, and technical system are common criteria for project decomposition.

An activity is a part of a project which has beginning and ending points, clearly identifiable, which are called events. It has a duration, which is the time interval elapsed between the moment the activity starts and the moment the activity is completed. It consumes resources of different types. It is allowed to start any time after its predecessors are completed.

Review of project scheduling methods for the resource-unconstrained case will be presented in section 3.2. Even though the assumption that resources are unconstrained is not a reasonable one, the concepts developed for this case are largely used in the resource constrained project scheduling methods, which are reviewed in section 3.3. Section 3.5 provides a review of generalized project scheduling methods. Emphasis will be given to the methods that generalize the scheduling problem in terms of resource application flexibility and stochastic elements added to the problem. Section 3.5.1 and 3.5.2 covers these two classes respectively.

3.2. Resource Unconstrained Project Scheduling Methods

There are two well known methods for planning and scheduling the project for the case that resources are unconstrained. These are CPM (Critical Path

† Other advantages for use of A-O-N representation are given by Moder and Philips(1964).

Method) and PERT (Program Evaluation and Review Technique). They both use similar network representations and activity scheduling. Since the two methods were developed independently , they bore the mark of environments for which they were created.

CPM was a result of a joint effort of DuPont Company and Remington Rand Univac Division originally aimed at better planning and control of the overhaul and maintenance of chemical plants. The project was started in 1957 and the CPM method was released in 1959 (Kelley and Walker).

PERT was developed 1957-1958 by a research team set up by the Navy Project Office. The methodology was published in 1959 (Malcolm et al.). PERT was developed for and has been used most frequently in aerospace industries, ie, R&D types of programs (Elmaghraby (1977)). These industries are relatively new; their technology is rapidly changing, and their products are non-standard. Thus PERT had to cope with the uncertainties that accompany R&D activities. Hence, PERT regards the total project duration as a random variable, and performs probabilistic calculations in order to characterize it.

CPM was developed in the context of routine operations whose durations were more or less well established. Therefore, CPM is basically deterministic. The major consideration of the originator of CPM was to reduce the total time during which the facility had to be shut down for overhaul.

3.2.1. Critical Path Method

The Critical Path Method is a technique for determining the "feasible" schedules (See Moder & Philips (1970) or Elmaghraby (1977).). The duration of each activity A_i, denoted by d_i, is assumed prespecified. It is also assumed A_i is not interruptible. Let $SS=(S_1, \ldots, S_N)$ be a schedule of activity start times.

Let A_j be a successor of A_i. If A_j may not start until A_i has finished the following inequality must hold:

$$S_i + d_i \leq S_j$$

Given all precedence relations relating the activity start times , the earliest start time , ES_i , of each activity A_i is computed by moving forward through the network. The earliest possible project completion time can be determined, ie, the earliest finish time of the last activity in the project. Using this date , or some specified completion time exceeding it, one can move backward through the network, thereby computing LS_i, the latest start time of each A_i. The *early schedule* is denoted herein by $ES=(ES_1, \ldots, ES_N)$ and the *late schedule* by $LS=(LS_1, \ldots, LS_N)$. Every feasible schedule S thus satisfies,

$$ES_i \leq S_i \leq LS_i \quad i=1,\ldots,N$$

The slack of activity i is defined as $TS_i = LS_i - ES_i$. It is the amount of time that the start of A_i can be delayed beyond ES_i if its successors operate late and its predecessors early. The activities with zero slack form the critical path(s) of the network.

3.2.2. PERT Method

As mentioned earlier, PERT and CPM use the same network representation and activity scheduling methods. PERT assumes activity durations are uncertain. For each activity in the project network, rather than trying to estimate directly the expected performance time of an activity, PERT developers chose to estimate three durations instead:

- The optimistic duration, o , is the shortest possible time to complete the activity, assuming most favorable conditions.
- The pessimistic duration, p , is the maximum time that would be necessary to perform the activity under most unfavorable conditions.
- The most likely duration, m.

To use those values one must specify the kind of probabilistic distribution associated with the activity duration. Clark (1962) proposed to use beta distributions for this purpose. His proposal has become extensively accepted. With this assumption, the estimated average duration, μ, and , estimated standard deviation of activity duration, σ are:

$$\mu = \frac{o + 4m + p}{6}$$

$$\sigma = \frac{p - o}{6}$$

Having the expected activity durations one can perform scheduling computations in section 3.2.1. For this case, we are mostly interested in a "forward pass" computation that yields the expected earliest start time of each A_i.

Some milestones, ie , important events, for the project may have preassigned scheduled occurrence times. Let S_i be the scheduled time for start of A_i. Then, knowing ES_i and S_i, the question is to what extent can we assert that the scheduled time will met. This is simply the problem of determining a confidence interval around estimated length of a particular path in the network. A path is a concatenation of activities. Therefore, its length, as the sum of uncertain durations, is itself a random variable. Probability theory results for sums of random variables enables us to estimate the probabilistic characteristics of path length, ie , path duration and variance.

In general PERT tends to yield optimistic estimates for expected path lengths and for the probabilities of achieving events by their schedule dates. This is mainly because the assumption that the critical path is fixed and given once for all is wrong in principle due to the variability in activity durations. An extensive investigation of this bias and of the assumptions underlying the PERT calculations has been made by Mac Crimmon and Ryavec (1961). Other authors like Clark(1961),Fulkerson (1962), Clingen(1967), Martin (1965), Elmaghraby (1967), and Robillard and Trahan(1977), to mention a few , have studied the problem in search of improved estimates of path length distribution functions. However, the dependency among path lengths renders the analytical treatment of correcting for the PERT bias quite difficult.

Monte Carlo simulation represents an alternative approach to this problem. Van Slyke (1963) uses this simulation procedure to establish the unbiased distribution of the project duration. In principle, each simulation runs consists of drawing at random, for each project activity, a duration time from the associated distribution function, and then computing the resulting project schedule. After a suitable number of runs, one can draw estimates of the mean and variance of the project duration.

3.3. Resource Constrained Project Scheduling Problem

Consider the problem of scheduling the activities within a project subject to resource use constraints. The literature addresses a number of related resource constrained scheduling problems (Herroelen (1972), Patterson (1973), Davis (1973), Cooper (1976), Elmaghraby (1977), Stinson (1976), Talbot (1982), Hax and Candea (1984)).

Time/Cost Tradeoff problem: The resource cost of performing each activity is expressed as a function of its duration. The problem is to find a schedule which minimizes the total cost of the project. Linear and convex cost-duration functions are commonly used in solution methods (Fulkerson (1961), Kelley (1961), Moder and Phillips (1970), Phillips and Dussouky (1977)). General more complicated discontinuous cost functions can also be handled for very small problems (Meyer and Shaffer (1963), Moder and Phillips (1970)).

Resource Loading Analysis: For a given project schedule, one produces time charts regarding the usage of various of resources required by the project activities. Charts may be used in checking against resource availabilities to establish whether the project is feasible. Or, an analysis of alternative loading profiles resulting from different project schedules may suggest ways to further improve resource utilization.

Resource Leveling: Costs are assumed to be a function of the peak use of each resource. Then, the problem is, given a prespecified project completion time and resource usage by activities, to find the schedule that minimizes the resource capacity costs. In general, as the combinatorial nature of the problem precludes optimal solutions, leveling procedures are heuristically based (Burgess and Killebrew (1962), Levy et al.(1962), DeWitte (1964), Moder and Phillips (1970), Leachman (1983)).

Resource Constrained problems: For given resource capacities, the project duration is to be minimized. Since the problem we try to deal with falls into this category, special attention will be given to this type of problem in the review.

3.4. Solution Methods for Resource Constrained Problem

In the resource constrained problem is, one allocates the limited resources to competing activities in order to achieve a schedule which results in the minimal project completion time or a schedule which has minimum cost. The problem assumes activities have fixed durations and fixed resource requirements.

The problem can be conceptualized by a mathematical formulation adapted from Talbot and Patterson (1978).

The following variables are used in the formulation:

F_i	Scheduled finish time for activity i; a project schedule is defined by a complete set of F_i .
d_i	Activity duration.
P_i	Set of immediate predecessors of activity i.
t	Time, assumed to be divided into integral periods, indexed by $t=1,\dots,T$.
A_t	Set of activities in progress at period t .
a_{ki}	Usage rate of resource type k by activity i.
R_{kt}	Amount of resource type k available in time period t .

For the objective of minimizing the project duration, the mathematical model is as follows:

$$\text{Minimize} \max_{i \in network} [F_i]$$

Subject to:

$$\max_{j \in P_i} [F_j] + d_i \leq F_i \quad i \in network$$

$$\sum_{i \in A_t} a_{ikt} \leq R_{kt} \quad k = 1, \ldots, K, \ t = 1, \ldots, T$$

The combinatorial nature of the problem is apparent if we consider A_t. Different project schedules may have different sets of A_t. A number of procedures are proposed to solve this problem. One can easily classify the solution methods into two types, one being *exact methods* (methods which produces optimal schedules), and the other being heuristic methods.

3.4.1. Exact Methods

Integer Programming and various Enumeration Methods are used to find an optimal solution to the problem.

Integer Programming

0-1 integer variables are used to identify the elements of the active activity set A_t, yielding an integer programming model. The problem is still hard to solve. In general the size of the problem that oan be handled by an integer program is found to be very limited (Pritsker at al. (1969), Wiest and Levy (1977)).

Enumeration Methods

One approach, that can easily be seen, is to enumerate all possible schedules, and select the best. However simple enumeration is computationally infeasible. A number of people have tried to cut the size of the enumeration either by introducing some heuristic that reaches the optimal solution faster, or, exploiting the special tree-like structure of the project scheduling problem which allows sequential decisions (Mueller-Mehrback (1967), Johnson (1967), Davis (1969), Davis and Heidorn (1971), Talbot and Patterson (1978)). Thus various implicit enumeration methods have been devised to find optimal schedules. These methods are still effective for only small problems. Reasonable computation times are being reported for networks with no more than 100 activities (Talbot and Patterson (1978), Patterson (1984)).

3.4.2. Heuristic Methods

Since the exact methods are ineffective for larger size projects, various heuristic procedures have been developed. Heuristic procedures can not guarantee optimality, but they are able to handle very large networks, and, in general, may give good enough schedules to serve the operational needs of users.

Heuristic methods are schemes for assigning activity priorities in making the activity sequencing decisions required for resolution of resource conflicts. There are a large number of heuristic procedures, some of which can be found in the open literature, while others are in the form of proprietary commercial computer software. The main distinguishing feature among heuristic methods is the measure that the heuristic uses to assign the priorities. A survey by Davis (1973) is given for the heuristic methods in the open literature.

Heuristics can be classified into parallel or serial routines, depending upon whether the priorities are assigned during scheduling or before any activities are scheduled, respectively. Although, there are some evidence that methods with parallel routines perform better than those using serial routines, there is no formal study that proves that superior methods always fall into one of the two categories.

A list of heuristic sequencing rules is given in Table 3.1. The table indicates the researchers who found the rules effective.

A study by Davis and Patterson (1975) provides a comparison of the relative effectiveness of various heuristic methods. The study covered the first seven rules in Table 3.1. Reported results show that the Minimum Slack rule (also called the Least Total Float rule) produced the best results for all performance parameters. The Longest Remaining Path and Minimum Late Finish rules performed close to the Minimum Slack rule. In a more recent study, Thesen (1976) introduced urgency factors to activities, which are calculated as linear combinations of activity resource utilization rates and the impacts of delays in the activity start times on the schedule. The urgency factors are used to find the set of starting activities for the scheduling period. A multi-dimensional knapsack problem which maximizes the sum of urgency factors of activities that can start in the scheduling period within resource limits, is solved at each scheduling period to find the starting activities. Thesen also reported that his

Priority Rule	Explanation	Researchers
Minimum Slack	Priority is given to the minimum slack activity $\underset{i}{Min}\,[LS_i - ES_i]$	Mize, Fendley, Patterson, Wiest
Longest Remaining Path	Priority is given to the activity with longest remaining series of activities. Calculations are made using precedence relations.	Verhines, Brand, Meyer, and Shaffer
Minimum Late Finish Time	Priority is given to the activity has minimum late finish time, ie, as calculated in CPM method.	Pascoe, Mueller-Mehrback, and Gounguet
Greatest Resource Demand	Priority is given to the activity that has maximum total resource requirements,ie, $= d_i \sum_k a_{ki}$	Knight and Patterson
Greatest Resource Utilization	Priority is given to the activities which results in maximum resource utilization at each scheduling period. This requires 0-1 integer programming.	RAMPS
Shortest Imminent Operation	Priority is given to the activity with shortest duration	Patterson
Most Jobs Possible	Priority is given to the combination of activities which results the greatest number of activities being scheduled in each scheduling period. This requires 0-1 integer programming.	Some commercial programs
Hybrid Rules	Priority is given to the activity which has highest score from the combinations of some of the previous rules	Thesen

TABLE 3.1. Summary of Common Priority Rules that are found in open literature.

approach was superior to the Minimum Slack priority rule according to his comparative experiments.

3.5. Generalized Resource Constrained Project Scheduling Problem

In this section, we will cover the methods that are used in solving the resource constrained problem with generalizations of some aspects of the common problem. Due to the our particular interest in the work characteristics in the shipyard, two kinds of generalized models will be reviewed. First, the methods that relax the assumption of fixed resource application rates and fixed durations will be reviewed. Second, the methods that handle the stochastic elements in project scheduling will be covered.

3.5.1. The Methods for Resource Application in the Generalized Problem

The first two published models which allow activities alternative rates of resource applications were Wiest's SPAR1(1967) model (Scheduling Program for Allocation of Resources) and RAMPS (Resource Allocation and Multi Project Scheduling) by Moshman et al. (1963). Both procedures consider that an activity may be performed either at a normal pace, at the fastest pace, or the slowest pace by using a normal, maximum, or minimum amount of resources, respectively. Although the RAMPS software has been widely distributed, the computational steps involved are nowhere fully described in detail.

SPAR1 basically assumes three discrete levels of resource applications for each activity (normal crew size, maximum crew size, and minimum crew size). An intermediate level can be applied to an activity only in special circumstances. The detailed procedure of the SPAR1 model will be covered in Chapter 7.

Weglarz (1981) studied properties of optimum solutions for doubly constrained scheduling problems when "performing speeds" of activities are continuous functions of resource applicatio... In his model, resources are not only constrained by a maximum usage rate at every moment but also by a maximum cumulative resource expenditure. The performing speed-resource usage function for each activity is a production function for the activity, relating the rate of completion to the rate of resource application. Although he generalizes the results for the cases when activities use more than one resource or have upper and lower bounds on their resource usage, it is not clear that the model works in the multi-resource case. He suggests a non-linear programming model to find the optimal solution for various generalized cases of the resource constrained problem (Weglarz (1976),(1979)). His solution technique is only applied to very small projects, as it involves modeling and computational difficulties for larger size problems.

Talbot (1982) introduced an integer programming formulation and a solution technique for the case that each activity has several alternative modes of accomplishment in the project. Each mode has a different magnitude and mix of resource usage. Detailed explanation of his method will be given in Chapter 7.

Leachman (1983) considered continuous ranges for activity intensity, ie, an index of the rate of resource applications to the activity, in the resource leveling problem. This continuous intensity range is adopted in the approach taken in Chapter 4 and 5.

3.5.2. The Methods that Handle the Stochastic Elements in the Project Scheduling Problem

Probabilistic project networking represents an extension of classical PERT in order to incorporate cases in which the way the project continues from some point on may depend on the uncertain outcome of one or several activities. Such situations are typical with R&D projects. One of the best known systems is GERT (Graphical Evaluation and Review Technique) originally developed by Pritsker and Happ (1966), Pritsker and Whitehouse (1966), and Whitehouse (1973).

GERT provides several flexibility features. It allows probabilistic branching, ie, the need for execution of some activities may be uncertain in the project network. It allows looping in the network, ie, depending on the outcome of an uncertain event, a branch of activities that has been executed before may be undertaken again. It allows alternative distributions for the probabilistic activity durations. (Recall that the PERT method assumes beta distribution for all the activities in the project.)

Usually GERT network analysis is carried out by simulation during which statistics on the project times and cost are collected. The GERT method basically concentrates on time-cost tradeoff studies, as it does not provide any treatment of resource constraints, resource application flexibilities, or uncertain resource requirements by activities.

VERT (Venture Evaluation and Review Technique) is an extension of GERT. It provides the capability to model decisions within the network in terms of times, costs, and performances. GERT can analyze times and cost only; hence VERT is a powerful project evaluation technique (Mueller (1972), Mueller and Digman (1981)).

4. MODELING ASSUMPTIONS OF A NEW SCHEDULING SYSTEM FOR THE PROJECT-ORIENTED PRODUCTION ENVIRONMENT

The review of project scheduling methods shows that no one of the methods alone proposed in the literature meets the needs arising from the characteristics of work in project-oriented production systems. So, a new approach of planning and scheduling the project in such systems is needed. In this chapter, the modeling assumptions for the new system will be put forward.

4.1. A Hierarchical Approach for Multi-Project Planning and Scheduling

A multi-project environment is assumed for the project oriented production system. In a multi-project environment where multiple projects might be executed at the same time, one could construct a large network by combining all of the project networks. Scheduling of activities would be done from the large network, utilizing total resources available in the production system. Howeve.', there are some disadvantages of this approach. The number of activities in the large network may become so large that the scheduling problem becomes intracable. In application, each project has different management. Scheduling of activities and allocation of resources between activities within a project is decided by the project's management. The approach introduced above requires scheduling and resource allocation decisions to be made by only one management for all projects. This may create some managerial difficulties.

An alternative to the above approach, namely a hierarchical project planning approach, can be used (R. Leachman and J. Boysen (1984).). In this approach, first, allocation of resources is made between projects. Then, preallocated resources to a project are used as capacities of resources for the project which will be allocated among project activities. The principal advantages of the hierarchical approach is its suitability to organizational structure and that it makes both multi-project planning and scheduling of individual projects easier to handle. Further advantages of hierarchical project planning and scheduling approach are given in R.C. Leachman and J. Boysen (1984).

4.2. Project Activities

An activity is a part of the project which has a beginning point and an ending point which are clearly identifiable. It has a duration. It requires resources of different types for its execution.

4.2.1. Precedence Relations between Activities

We shall consider the case when precedence relations between activities may be either strict or overlapping. Overlapping activity precedence may be used in reducing the network size. The precedence of this type is frequently useful in the networks modeling ship overhauls. So, project networks considered here include both types of precedence relations.

4.2.2. Flexible Resource Applications for Activities

We shall assume that all the different types of resources which are required by an activity are applied proportionally through the activity execution. In other words, each activity utilizes a constant mix of resources as it progresses. When the rates of application of different types of resources to the activity are proportional, the rates can be indexed in terms of one rate which will be called the intensity of the activity. The rate of resource application of an activity (activity intensity) is assumed to be variable within upper and lower limits. Each activity is assumed to have a normal resource application rate and a normal duration corresponding to constant normal intensity. For a lower rate of resource application, activity duration will be longer or, for a higher level of resource application, activity duration will be shorter.

The flexibility in resource application to activities has significant importance in project scheduling. The scheduling problem in the case of fixed resource application rates and fixed durations is just to decide starting times of project activities which will not violate resource constraints. For flexible resource application rates to activities, the problem requires deciding resource application rates of activities in addition to their start times.

4.2.3. Stochastic Resource Requirements For Activities

Resource requirements of an activity are best viewed as random variables with some distributions in project-oriented production systems. This is also true for construction projects but the projects undertaken by construction firms are often not similar to one another. When projects are different, collection of historical resource requirement data for activities is not possible. Projects are similar or even identical in project-oriented production systems. Projects may be executed a large number of times in such systems. Consequently, it may be possible to collect data necessary for finding distributions of resource requirements of activities.

4.3. Resources Used In Projects

In general, two types of resources are used in projects. These are,

a) Non-storable Resources

Example: Trade labor of different types, machine or equipment services used in production.

b) Storable Resources

Example: Money, material, etc.

Our main concern will be given to scarce non-storable resources such as trade labor from different shops. Project-oriented production systems may include a large number of trade labor "shops". Each shop consists of one type of skilled trade labor. Employees in these shops work on a fixed salary basis. The cost of labor to the production system is mostly fixed, perhaps with the possibility of incremental overtime.

Each shop has a capacity at a certain time, ie, a number of man-hours available per day. Capacity may change with time but capacity available in a given time period is inflexible. The total shop capacity is shared by the ongoing projects in the system. The aggregate planning level allocates the shop capacities between projects as discussed earlier. Thus each project has preallocated capacity from all shops. This capacity is to be allocated between activities of the project that are ready for execution at that time.

Project cost for the production system largely depends on project time. The objective of scheduling in such a system is to finish the project as early as possible within given resource capacities. Equivalently, the objective is to maximize the throughput , ie, the number of projects completed by the system within a given time horizon.

Storable resources may be limited for usage per period as non-storable resources are, and also, they may be limited in their cumulative consumption over the period of project duration. The latter type of resources is of secondary concern in our case, so that the method of scheduling is first designed assuming that we have no limitations on the cumulative consumption of resources. The extension of the method with necessary adjustments to handle cumulative resource consumption limitations will be given in Appendix 2.

4.4. Stochastic Project Networks

In projects like overhauling a ship, building a ship or an airplane, advanced testing of subsystems is needed following their production or repair. For subsystems involving high safety requirements, eg, nuclear power generating system in nuclear submarines, testing and repair rework must be undertaken until performance of the subsystem achieves very tight tolerance limits. This may take a number of test-rework cycles. The number of test-rework cycles for the same test in different examples of similar projects may vary greatly.

We refer to activities involving testing of subsystems as test activities. The outcome of a test activity can be a success or a failure. If it is a success then its regular follower activities may start. If it is a failure, depending on the rework type needed, one of the possible small subnetworks of rework activities must be executed before the test activity may be undertaken again.

Assuming that all the test activities will be successes on the first trials, we have a basic deterministic network structure. After each test there are a number of small subnetworks, one of which is to be executed after a failure depending on the type of the failure. Consequently, a special stochastic network structure is assumed for the projects.

5. MODEL WITH VARIABLE INTENSITIES FOR ACTIVITIES

As discussed before, there is considerable flexibility available to management in loading of manpower resources on each activity of a project. Most production activities may be progressed with varying intensity, ie, varying levels of manpower application. In this chapter, we develop the model which allows variable intensities for activities.

5.1. Linear Activity Analysis Model

We suppose a project is characterized by an activity-on-node critical path analysis network of N activities, denoted by $A_1, A_2, A_3, \ldots, A_N$. The project requires utilization of K infinitely divisible, non-storable resources, denoted by $R_1, R_2, R_3, \ldots, R_K$. Following Shephard and Mak (1982) and Leachman (1983), the rates of application of various resources to a given activity A_i are assumed to be proportional and constant on discrete time intervals (0,1],(1,2],...,(t-1,t],... . For convenience, we refer to the time interval (t-1,t] as *period t*. The rates of resource applications to activity A_i in period t are indexed by an *intensity function* $z_i(t)$, taken with technical coefficients as follows:

a_{ki} the total amount of exogenous resource k that is required by A_i from its start to the end , k=1,2,...,K .

To illustrate this model, if an activity A_i is operated continuously at a constant intensity z_i , then its duration is $1/z_i$, and resource R_k is loaded at the rate of $a_{ki}z_i$ per unit time over the duration. Note that most treatments of resource loading based on ordinary critical path analysis implicitly assume activities are operated in this constant loading fashion.

We shall assume each activity A_i has a ***normal duration*** d_i which corresponds to normal application rates of resources to the activity, applied continuously until it is complete. Then, normal intensity is defined as,

$$z_i = \frac{1}{d_i} \tag{5.1}$$

With this assignment, the intensity function $z_i(t)$ corresponds to the fraction of the activity done in period t. So, summation of $z_i(t)$ between start and finish must add up to unity, ie,

$$\sum_{t=t_s+1}^{t_f} z_i(t) = 1. \tag{5.2}$$

where t_s & t_f are the start and finish times of A_i.

The intensity of each activity A_i is assumed to have upper and lower bounds corresponding to practical minimum and maximum application rates of resources. That is, from the start of the activity until its completion, activity intensity lies in the range

$$\underline{z}_i \le z_i(t) \le \overline{z}_i \tag{5.3}$$

where the bounds $\underline{z}_i$ and $\overline{z}_i$ are given†. Before the start and after the completion of A_i , $z_i(t)$ is zero.

5.2. Intermediate Activity Transfers

In critical path analysis, activities are usually related by strict precedence. Occasionally, in order to reduce the number of the activities required, overlap

† In the case that resource capacities decrease with time, activities may be disrupted, in which case activity intensity is forced to zero for a period between start and completion of the activity. Disruption of activities occurs only as last recourse in the heuristic algorithms to be developed.

relationships may be specified. (See Elmaghraby (1977).). As an example, suppose A_j may not start until after A_i has performed a fraction f_{ij} of its work. If S_i is the start time of A_i, and S_j is the start time of A_j, then S_j is constrained by

$$S_i + f_{ij}\, d_i \leq S_j \tag{5.4}$$

Note that $f_{ij}=1$ corresponds to strict precedence and $f_{ij} d_i$ is approximated to closest integer value.

When activity intensities are variable, the overlap relationship (5.4) must be generalized. To do this, we shall use the concept from production theory of intermediate product transfer between dependent (ie, arc-connected) activities, as set forth in Shephard, Al-Ayat and Leachman (1977), and Leachman (1982).

We shall say that there is a progress lag of f_{ij} between A_i and A_j, ie, A_i must be $100(f_{ij}+\delta)\%$ before A_j can be $100\delta\%$ complete. Formally, given an f_{ij} satisfying $0<f_{ij}\leq 1$, we have , for all t,

$$\sum_{\tau=1}^{t} z_j(\tau) \leq \begin{cases} Max\left\{0, \sum_{\tau=1}^{t} z_i(\tau) - f_{ij}\right\} & \text{if } 0\leq \sum_{\tau=1}^{t} z_i(\tau) < 1.0 \\ 1.0 & \text{if } \sum_{\tau=1} z_i(\tau) = 1.0 \end{cases} \tag{5.5}$$

Once again, $f_{ij}=1$ corresponds to strict precedence.

In the case that intensity must be held constant at unity from start to finish of each activity, (5.5) holds for all t if and only if (5.4) holds and also,

$$S_i + d_i \leq S_j + (1-f_{ij})\, d_j \tag{5.6}$$

Note that if $d_i \leq d_j$, then (5.4) implies (5.6). However, for the progress lag here, if $d_i > d_j$, it is not sufficient to simply separate the start times of constant intensity activities A_i and A_j according to the lag; the activities must also have finish times separated according to the lag, as expressed in (5.6). When intensity is allowed to vary, the more general expression (5.5) applies.

When A_i and A_j are related by a progress lag, we shall say there is a *flow transfer* between them. If we view A_i as producing intermediate product required as input by A_j then after an initial lag, A_i continuously supplies intermediate product to A_j, allowing progress to be made towards completion of A_j. When $f_{ij}=1$, there is a strict precedence between A_i and A_j ,ie ,the transfer of intermediate product is entirely event-based. We shall assume in the following that the activity network consists of both event-based and flow transfers of intermediate products.

6. ALGORITHMS FOR INTENSITY ASSIGNMENT

6.1. Introduction

This chapter presents two algorithms (Upgrading-Only and Upgrading and Downgrading) for resource constrained project scheduling with variable intensities for activities. An intensity assignment algorithm can be visualized as a decision maker which decides the intensities of activities which will be maintained until the next decision time. Through intensity assignment to an activity, resources† used by this activity are also allocated, as explained in section 5.1. At each decision time, the algorithms allocate the resources up to their capacities among activities that are already started or ready to start at that time. Decisions have to be revised at time points when there is a change in the system that may effect the intensity assignment. These changes are completion of an activity, the event of an activity becoming ready to start, and a change in capacities of resources. For the moment, it is assumed that all resource capacities are constant. This assumption will be relaxed in section 5.5.

The main objective of the scheduling problem is to find a schedule which gives the least project completion time within the capacity restrictions and event due dates which are determined by the aggregate plan. The intensity assignment algorithms to be developed consist of 'clever' heuristic decision rules to be applied at each decision time. Consistent application of these heuristics will result in a schedule with a 'good' project finish time.

† Only non-storable resources are considered in this chapter. The extension of the methodology introduced in this chapter for storable resources will be given in Appendix A2.

The basic intuition behind the logic of both algorithms is ***equalizing relative lateness (or earliness)*** of activities against their target finish times through assigning appropriate intensities to the activities, argued as follows.

A Late Finish Time of an activity can be visualized as a due date for the activity finish, if it is not specified by the aggregate plan, such that if the activity finishes any later than the due date, it will lengthen the project completion time or violate the aggregate plan. Since this is true for all activities, the decrease in project finish time can only be achieved if all activities finish early with respect to their due dates. Any one activity being late with respect to its due date is enough to increase the project finish even if all the others are early with respect to their due dates. The sensible objective at each decision time is to make all activities , if possible, to be as early as possible with respect to their target late finish times. This is accomplished by equalizing earliness.

6.1.1. Activity Target Finish Time

The target finish time of an activity is taken as the Late Finish Time of the activity as it is computed using normal durations in ordinary critical path calculations subject to the due dates specified by the aggregate plan. Before each intensity assignment, the critical path Late Finish Time calculation is performed for the unexecuted part of the activity network, ie, the activities that are not started yet and the activities that are started but not finished yet. In this calculation, it is assumed that activities will use constant normal intensities between their start and finish times, ie, the same assumption as in critical path analysis. For the activities that are already started and already have intensities assigned, normal intensity is assumed for their uncompleted fractions.

6.1.2. Notation

We shall use the following notation (in addition to notation set forth in Chapter 5).

A_i	set of indices of activities to which A_i supplies intermediate product (followers).
Γ_i	set of indices of activities supplying A_i with intermediate product (predecessors).
t	current time period, ie, reallocation of resources is considered at time $(t-1)$.
$x_i(t)$	intensity of A_i in time period t .
S_i	actual starting time of A_i.
F_i	actual finishing time period of A_i.
$F_i(t)$	finishing time of A_i projected at time t .
$FT_{i,l}(t)$	flow transfer time for follower A_l ,ie, the time that activity i will be progressed enough to allow its immediate follower l to start, projected at time t .
$pcv_i(t)$	fraction of A_i completed at time t . ($= \sum_{\tau=1}^{t} x_i(\tau)$)
$LF_i(t)$	target late finish time of A_i computed at time t , computed by an ordinary (ie, resource-unconstrained) critical path analysis backward pass

assuming normal intensities for all activities. Detailed calculation of $LF_i(t)$ will be described in section 6.1.3.

$lsc_i(t)$	score of lateness of A_i at time t ,ie, lateness with respect to its target late finish time.
$esc_i(t)$	score of earliness of A_i at time t ,ie, earliness with respect to its target late finish time.
$r_k(t)$	capacity of resource R_k in time period t .
$ar_k(t)$	amount of resource R_k available, ie, unallocated, in time period t .

IACTIVE set of indices of currently active activities, ie,

$$\left\{ i \mid 0. < pcv_i(t-1) < 1.0 \right\}$$

IREADY set of indices of activities that are currently ready to start ie, those activities not yet active whose predecessors are progressed enough to satisfy the progress lag (equation 5.5).

6.1.3. Late Finish Updates

Recalculations of late finish times of activities are made before every assignment of intensities, that is, there is a recalculation of the times used as targets for activities for intensity assignment purposes. Recall that parallel routines update the priorities in the same way for the fixed intensity case. In calculating these times, it is assumed that activities which are not finished at the time of update, will run with normal intensity from that time on. This is true for active activities, regardless of current intensity level, and also for unstarted activities. The following steps are performed at each late finish update.

1. Find Adjusted Finish Times for all active and completed activities (A_i). The projected finish time $F_i(t-1)$ reflects the current intensity of A_i ,ie, $x_i(t-1)$. $AF_i(t-1)$ is the adjusted finish time of A_i assuming that it will have unit intensity after time $(t-1)$. It is calculated as follows:

   (a) If an activity is finished at or before the current time $t-1$,ie, $t-1 \geq F_i$, then

   $$AF_i(t-1) = F_i$$

   (b) If the activity is in progress, then

   $$AF_i(t-1) = t-1 + (\ 1.0 - pcv_i(t-1)\ )\ d_i$$

   (c) If the activity is not started yet, then set the adjusted finish time to a big number , T , so that it is distinguished in future calculations.

   $$AF_i(t-1) = T \quad \text{if} \quad t-1 < S_i$$

2. Run CPM type calculations to find late finish times as follows:

   (a) Set the early finish times of activities that are finished or started but not finished at time $(t-1)$.

   $$EF_i(t-1) = AF_i(t-1) \quad \text{if} \quad AF_i(t-1) \neq T$$

   where $EF_i(t-1)$ is early finish time of A_i at time t-1 .

   (b) Make forward calculations to find Early Finish times for unstarted activities, ie, for $i = 1,2,\ldots,N$ and A_i unstarted,

   $$EF_i = \underset{j \in \Gamma_i}{Max}\,[EF_j] + d_i$$

(c) Make backwards pass to find target late finish time ($LF_i(t)$).

(i) For each activity A_i that has a due date (DD_i) specified by the aggregate plan, set its Late finish time equal to the due date.

$LF_i(t) = DD_i$

(It is assumed that all activities with no followers have due dates specified by the aggregate plan. Other activities may have due dates as well.)

(ii) Make backward calculations for the activities that have followers. For $i = N, N-1, \ldots, 1$ such that A_i is nonempty, set

$LF_i(t) = \underset{j \in A_i}{Min} \, [LF_j - d_j]$

6.1.4. Equalizing Relative Lateness (or Earliness)

As discussed before, the ideal objective of intensity assignment is equalizing earliness. The problem can be formulated mathematically as follows.

Let $S = A_1, \cdots, A_i, \cdots, A_L$ be the set of activities that are candidates for intensity assignments at time $t-1$.

Then we wish to

$$Min \sum_{i=1}^{L} (\, F_i(t) - LF_i(t-1) \,)^2$$

subject to:

$$F_i(t) = \frac{(\, 1 - pcv_i(t-1) \,)}{x_i(t)} + (t-1) \qquad for\ all\ A_i \ \varepsilon\ S \quad (\text{ Finish Time })$$

$$\sum_{i=1}^{L} a_{ki} \, x_i(t) \le r_k(t) \qquad for\ all\ k \ (\text{Resource Availablity})$$

$pcv_i(t-1)+x_i(t) \leq pcv_k(t-1)+x_k(t)-f_{ki}$ for $k \varepsilon \Lambda_i$, $A_i \varepsilon S$, $A_k \varepsilon S$ (Flow transfers)

$\underline{x}_i \leq x_i(t) \leq \overline{x}_i$ or $x_i(t) = 0$ for *all* $A_i \varepsilon S$ (Intensity bounds)

Note : $x_i(t)$'s and $F_i(t)$'s are decision variables. Other parameters have known values at time t.

The above formulation has a quadratic objective function with some integer variables used to handle the complication caused by the fact that $x_i(t)$ has to be either within its upper and lower bounds or zero. The problem does not have a structure that admits an easy solution methodology to find the optimal solution.

A method for a simplified version of this problem is solved by Bitran and Hax (1981) for the problem of ***equalizing run-out times*** of inventories of the items in the same product family in a hierarchical production planning system. In their model, there is only one capacity constraint and decision variables have only upper and lower bounds. Their method is not applicable to our problem with multiple capacity constraints. Consequently, we shall develop a greedy priority approach to allocate resources to meet target finish times as discussed below.

6.2. Greedy Method of Assigning Intensities to Activities.

At each decision time a list of activities which are candidates for intensity assignment is generated. The relative lateness considering target Late Finish Times of the activities on the list is assessed. Activities on the list are prioritized using lateness scores equal to $(F_i(t)-LF_i(t))$, whereby the larger the lateness score, the higher the priority. These priorities determine the order that activities will be considered for intensity assignment. Note that, the priority rule selected is similar to the 'Minimum Slack' and 'Earliest Late Finish' rules tested and found superior (in the fixed intensity case) to other rules by Davis and Patterson (1975), and others†.

Three different passes are made before finalizing the intensity assignments to the activities in the candidate list. In the first pass, using the priority list, activities are considered one by one until all of them are considered. For each activity, an attempt is made to upgrade the intensity of the activity (or start the activity), if necessary, up to the critical intensity level that will allow the activity to just meet its target finish time. Note that resource availabilities, the intensity upper bound, or progress lag limitations allowed by flow transfers may be more restricting than the critical intensity level. If this is the case, the most restricting level is taken. If the intensity level taken is under the lower intensity bound of the activity than zero intensity is assigned to the activity, ie, the activity remains unscheduled, at this pass.

In the second pass, a look ahead-procedure is used to correct mistakes that may be made in earlier intensity assignments. The procedure is similar to the one used in Wiest's SPAR1 model. In this pass, attempts are made to upgrade intensities of the *critical* †† and not yet started activities by rescheduling (ie, delaying) the *truly slack* ††† activities started in earlier periods. The only reason a critical activity would not be started at the first pass is if

† Mize (1964), Pascoe (1965), Gonguet (1969), Patterson (1973).
†† An activity is called critical if it can not catch up to its late finish time running with normal intensity from the current time t, ie, its lateness score is negative.
††† An active activity is called truly slack if by starting the activity in period (t+1) with normal intensity, it would be completed before its target late finish time.

resource availability for the activity (or its predecessors) is restricting. By rescheduling the truly slack activities, the resources which were tentatively allocated to these activities are made available to such critical activities.

In the third pass, using the priority list, activities are again considered one by one until all activities are considered. For each activity, an attempt is made to upgrade the intensity of the activity up to its upper intensity bound. Again resource availability or progress lag limitations of flow transfers may be restricting. The most restrictive level of intensity is assigned to the activity, if this level is higher than the intensity lower bound of the activity. Otherwise, zero intensity is assigned to the activity.

Intensities assigned to activities at the third pass are the levels that activities will operate at until next intensity assignment (ie, decision) time. If resources are tight, activities which are at the top of the priority list will be allocated resources, while activities which down in the list will be postponed. Eventually, all activities so postponed become critical and move to the top of the priority list.

Using this basic approach, two algorithms which reflect two different management policies concerning resource reallocation are introduced in sections 6.3 and 6.4.

6.3. Upgrading-Only Algorithm

We first consider the case where management does not allow reduction in the level of resources which is already committed to an activity once it is started, although it allows increases in this level while the activity is in progress. The Upgrading-Only algorithm is designed to cope with this case. This algorithm reassigns the resources to the activities by reassigning intensities to the activities, and, it is employed whenever there is an event that may cause a change in current resource allocation. The algorithm is initialized as follows.

INITIALIZATION:

$IACTIVE = \Phi$

$IREADY = \left\{ i \mid \Gamma_i = \Phi \right\}$ for *all* i

$x_i(t) = 0.$, $pcv_i(t) = 0.$ for all t.

$ar_k(1) = r_k(1)$ for all k.

The first event is the start of the project, time $t=0$. Follow-on events are calculated by the algorithm. When an event that may cause resource reallocation occurs, the algorithm performs the following steps:

STEP−1

Update target late finish times $LF_i(t-1)$ by using the procedure described in section 6.1.3.

Note: time (t-1) denotes the time just before resource allocations, and period t denotes the period $(t-1,t)$ in which allocations are applied.

STEP–2 Upgrade if necessary intensities of active and ready activities so that they may catch up to their late finish times (targets), as follows.

(a) Update Lateness Scores assuming ready activities will run at normal intensity.

$lsc_i(t-1) = F_i(t-1) - LF_i(t-1)$ for all $i \in IACTIVE$

$lsc_i(t-1) = (t-1) + d_i - LF_i(t-1)$ for all $i \in IREADY$

(b) Prioritize activities in combined list (*IREADY* and *IACTIVE*) by sorting them in decreasing order with respect to Lateness Scores. In the case of ties, any order suffices which is consistent with network logic.

(c) Upgrade intensities of activities in combined list one by one according to priority as follows:

set,

$$y_i(t) = Min \begin{cases} \bar{z}_i & \textit{Intensity Upper Bound} \\ \min\limits_{k} \left\{ \dfrac{ar_k(t)}{a_{ki}} \right\} + z_i(t-1) & \textit{Resource Availiblity} \\ (1.0 - pcv_i(t-1)) \dfrac{1.0}{LF_i(t-1) - t} & \textit{Target Finish Time} \\ \min\limits_{j \in \Gamma_i} \left\{ (1.0 - pcv_i(t-1) - f_{ji}) \dfrac{1.0}{F_j(t-1) - (t-1)} \right\} & \textit{Flow Transfer}^{\dagger} \end{cases}$$

Note that $y_i(t)$ is a temporary variable to calculate $z_i(t)$, which is defined in terms of $y_i(t)$ as follows:

$$z_i(t) = \begin{cases} y_i(t) \text{ if } & y_i(t) \geq z_i(t-1) \text{ and } y_i(t) \geq \underline{z}_i \\ z_i(t-1) \text{ if } & y_i(t) < z_i(t-1) \text{ and } z_i(t-1) \geq \underline{z}_i \\ 0 \text{ if } & y_i(t) < z_i(t-1) \text{ and } z_i(t-1) < \underline{z}_i \end{cases}$$

† Derivation of this constraint is given in Appendix A1.

Then set

$$F_i(t) = t-1 + (1.0-pcv_i(t-1))\frac{1.0}{x_i(t)}$$

and reset

$$ar_k(t) \leftarrow ar_k(t) - a_{ki}(x_i(t)-x_i(t-1))$$

Repeat the above for $i \in$ *IACTIVE* $\cup$ *IREADY* with next highest priority, until all the activities in combined list are processed. Then, continue to step 3 .

STEP–3 Try to upgrade intensities of the *critical* and not yet started activities by rescheduling the *truly slack* activities started in earlier periods. The only reason a critical activity would not be started at step-2 is if the available resources are not enough to start the activity with an intensity level which is higher than its lower intensity bound. By rescheduling the truly slack activities, the resources which are used by these activities are made available to such critical activities.

This is done as follows:

(a) Find a critical but not started activity from ready activity list (*IREADY*). If there is not such an activity, or if we have attempted all to upgrade all such activities continue to step-4. Otherwise, let A_i be such an activity. Then it satisfies

$lsc_i(t) \leq 0$ and $x_i(t-1) = 0$ where $i \in$ *IREADY*

(b) Determine if enough resources will be obtained by rescheduling truly slack activities that are already started.

(i) Initialize the search variables:

$rneed_k = a_{ki}\,x_i - ar_k(t)$ for all k

$IRESCHED = \Phi$

Note that $rneed_k$ and *IRESCHED* are temporary variables. $rneed_k$ represents further resource needed from resource type k to start A_i. *IRESCHED* represents the list of activities that are to be rescheduled in order to start A_i.

(ii) Find a truly slack activity from the active activity list (*IACTIVE*). Let A_j be such an activity. Then it satisfies:

$t + d_i \leq LF_j$

If there is such an A_j, continue to (iii).

Otherwise, go to (a) to get another critical activity.

(iii) Check if the resources needed by A_i have been obtained.

Update:

$rneed_k \leftarrow rneed_k - a_{kj}x_j(t-1)$ for all k

then,

If $rneed_k \leq 0$ for all k then continue to (c)

Otherwise, put A_j into the reschedule list.

$IRESCHED \leftarrow IRESCHED + j$

Go to (ii) to find another activity to reschedule.

(c) Reschedule the activities in the list *IRESCHED* since they provide enough resources to start A_i, and upgrade the intensity of A_i.

(i) Update:

$ar_k(t) \leftarrow ar_k(t) + a_{kj}\, z_j(t-1)$ for all $j \in$ *IRESCHED*

$z_j(t) = 0$ for all $j \in$ *IRESCHED*

$pcv_j(t) = 0$ for all $j \in$ *IRESCHED*

IACTIVE $\leftarrow$ *IACTIVE* $- j$

IREADY $\leftarrow$ *IREADY* $+ j$

(ii) Upgrade the intensity of A_i. This can be done by following the same procedure as in step-2 (c).

(iii) Go to (a) to find another critical activity.

STEP−4 Use remaining resources for further upgrading of activities in combined list.

(a) repeat step 1(a)

(b) repeat step 1(b)

(c) Upgrade Intensities using same rule in 1(c) except revise $y_i(t)$ as follows:

$$y_i(t) = Min \begin{cases} \bar{z}_i & \textit{Intensity Upper Bound} \\ \min\limits_{k} \left\{ \dfrac{ar_k(t)}{a_{ki}} \right\} + z_i(t-1) & \textit{Resource Availiblity} \\ \min\limits_{j \in \Gamma_i} \left\{ (1.0 - pcv_i(t-1) - f_{ji}) \dfrac{1.0}{F_j(t-1)-(t-1)} \right\} & \textit{Flow Transfer} \end{cases}$$

STEP−5 Make necessary updates:

(a) Record the start of each new activity: if A_i is such that $x_i(t-1)=0$ and $x_i(t)\geq 0$ then,

$$IREADY = IREADY - i$$

$$IACTIVE = IACTIVE + i$$

(b) Update flow transfer times of upgraded activities

$$FT_{i,l}(t) = (f_{il} - pcv_i(t-1))\,\frac{1.}{x_i(t)} \qquad \text{for } l \in \Gamma_i$$

(c) Find next resource allocation time t_n:

$$t_n = Min\left\{F_i(t)\,,\,FT_{i,l}(t)\right\} \qquad \text{for } i \in IACTIVE \text{ and } all\ \ l \in \Lambda_i$$

Note: resource allocation is needed either when an activity finishes or when new activities become ready to start. If calculated t_n is decimal, it will be rounded. As discussed in section 6.5.2, if there is a capacity change before time t_n, the time of capacity change is treated as the next event t_n.

(d) Update intensities and cumulative intensities during (t,t_n)

$$x_i(\tau) = x_i(t)\,, \qquad \tau = t+1,\ldots,t_n$$

$$pcv_i(\tau) = pcv_i(\tau-1) + x_i(t)\,, \qquad \tau = t,\ldots t_n$$

$$F_i(\tau) = F_i(t)\,, \qquad \tau = t+1,\ldots,t_n$$

$$FT_{(i,l)}(\tau) = FT_{i,l}(t)\,, \qquad \tau = t+1,\ldots,t_n$$

(e) Update ready activity list:

$IREADY \leftarrow IREADY + j$ for all j satisfying $pcv_l(t_n) \geq f_{lj}$

for all $l \epsilon \Lambda_j$

(f) Update available resources to reflect activities finishing at t_n:

$ar_k(t_n) = ar_k(t) + a_{ki}\, x_i(t)$ if A_i is such that

$pcv_i(t_n) = 1.$ and $pcv_i(t) > 1.$

(g) Update current resource allocation time:

$t \leftarrow t_n$

STEP–6 Check if all activities are finished:

If $IREADY = IACTIVE = \Phi$, then all activities are done,

ie, end of scheduling.

Otherwise go to step 0 for resource reallocation.

6.4. Upgrading and Downgrading Algorithm.

We now consider the case where management allows both increase and decrease in the level of resources which is assigned to an activity that is in progress. The Upgrading and Downgrading algorithm is designed to cope with this case.

The Upgrading-Only algorithm is easily modified for this case. We need to perform a step before the Upgrading-Only algorithm which is as follows.

STEP−0

Set (tentatively) the intensities of $i \in IACTIVE$ to their lower intensity bounds, as follows:

$$x_i(t) = \underline{x}_i \qquad \text{for } i \in IACTIVE$$

$$F_i(t-1) = (t-1) + (1.0 - pcv_i(t-1)) \frac{1.0}{\underline{x}_i}$$

and reset

$$ar_k(t) \leftarrow ar_k(t) + a_{ki}(x_i(t) - x_i(t-1))$$

Then perform the steps of the Upgrading-Only algorithm.

As can be seen, all active activities are tentatively set to lower intensity bounds in order to increase pools of available resources to maximum levels. The activities are then upgraded according to urgency in steps 1 through 6 of the Upgrading-Only algorithm.

6.5. Handling Time-Varying Capacities For Resources.

6.5.1. Basic approach

In project-oriented production systems resource capacities available to a particular project change frequently. A change in resource capacities is one of the events that may require changes in the intensities of the activities assigned using one of the algorithms. If at a decision time there are increases in the capacities of certain types of resources, the intensity assignment algorithms would be used to reassign the intensities of the activities in order to utilize the extra resources available from the capacity change. If there are decreases in capacities of some resource types, withdrawal of resources from some activities may be necessary. Even stopping the executions of some activities may be necessary but stopping activity execution is considered only as a last resort.

A heuristic procedure is applied in each decision time with capacity changes. The heuristic tries to restore the feasibility from infeasibilities which may be caused by resource capacity changes. If there is no infeasibility, ie, new capacities for the resources are greater than the total resources used by all activities which are in execution, then intensity assignment algorithms are used to reallocate the resources between activities. If there are infeasibilities, ie, new capacities of some of the resources are smaller than the total resources used by the activities which are currently in execution, then attempts to restore feasibility will be made.

A priority list is generated for the activities that are active using their earliness scores, ie, the negatives of their lateness scores. First, using this priority list, activities are considered for intensity decrease, one by one, until all infeasibilities are eliminated or until all activities are considered. Intensity of each activity is set to its lower bound or to an intensity level that will be just enough to restore feasibility. Note that if an activity does not use the resource types that are causing infeasibility its intensity remains same. If all infeasibilities are not eliminated in this step, then, using the same priority list, activities are considered for disruption until feasibility is restored.

6.5.2. Capacity Change Procedure

A time of capacity change is treated as an event. We now designate capacities as $r_k(t)$, k=1,...K . Suppose for some k , $r_k(t-1) \neq r_k(t)$,ie , there is a capacity change event at time t-1. For such an event, the following steps are performed by the algorithm before the steps of the intensity assignments at that event. Let $\Delta r_k(t)$ be amount of capacity change for resource type k at time t, ie, $\Delta r_k(t)=r_k(t)-r_k(t-1)$.

STEP−1 Update following parameter,

$$ar_k(t) \leftarrow ar_k(t) + \Delta r_k(t) \qquad\qquad \text{all } k$$

STEP−2 Check if there are any capacity violations

If $ar_k(t) \geq 0$. then Go to 5

Otherwise Go to 3

STEP−3 In this step, we downgrade intensities in an attempt to rectify capacity violations.

(a) Update target late finish times $LF_i(t-1)$ by using the procedure described in section 6.1.3.

(b) Calculate Earliness Scores for all active activities.

$$esc_i(t-1) = LF_i(t-1) - F_i(t-1) \quad \text{for } i \in IACTIVE$$

(c) Prioritize activities by sorting active activities in decreasing order with respect to Earliness Scores. In the case of a tie any order is acceptable which is consistent with network logic.

(d) Downgrade intensities of active activities one by one according to priority as follows. Choose the active activity with the highest earliness score, and set

$$z_i(t) = \begin{cases} z_i(t-1) & \text{if } a_{ki}=0 \text{ for } all\ k\ such\ that\ \ ar_k(t)<0 \\ Max\left\{\left[z_i(t-1) - \underset{k}{Min}\left\{\dfrac{-ar_k(t)}{a_{ki}}\right\}\right], \underline{z}_i\right\} & otherwise \end{cases}$$

Update,

$$F_i(t-1) = t-1 + (1.0 - pcv_i(t-1))\,\frac{1.0}{z_i(t)}$$

$$ar_k(t) \leftarrow ar_k(t) + a_{ki}(z_i(t) - z_i(t-1))$$

If $ar_k(t) \geq 0$ for all k ,then , go to step 5 : otherwise repeat the above routine for each $i \in IACTIVE$ in order of highest priority. If all active activities have been processed, continue to step 4.

STEP–4 In this step, activities are disrupted until capacity violations are eliminated. Retaining the prioritization from step 3, activities are disrupted one by one according to priority, as follows.

$$x_i(t) = \begin{cases} x_i(t-1) \text{ if } a_{ki}=0 \text{ for } \textit{all } k \textit{ such that } ar_k(t) > 0 \\ 0. \qquad \textit{otherwise} \end{cases}$$

Update,

$ar_k(t) \leftarrow ar_k(t) + a_{ki}\,(x_i(t)-x_i(t-1))$

$F_i(t-1) = T$ $\qquad$ (T is a big number)

If $a_{ki}(t) \geq 0$, go to step 5, otherwise repeat the above for $i \epsilon IACTIVE$ with next highest priority.

STEP–5 Go to Intensity Assignment Algorithm

7. VALIDATION OF INTENSITY ASSIGNMENT ALGORITHMS

The aim of this chapter is to provide some evidence of validity of the intensity assignment algorithms which are introduced in Chapter 6. The best way of doing this would be to compare the schedules that are obtained from intensity assignment algorithms to the optimal schedule, if there was a method available which provides an optimal solution to the problem. Unfortunately, such a method does not exist. Another way would be to compare the schedules that are obtained from intensity assignment algorithms to the schedules that are obtained from some other methods which solve the problem with the same assumptions that are made for intensity assignment algorithms. Unfortunately, no methods, that we know, have been offered in the literature that solve the problem with exactly the same assumptions. But, two methods which solve the problem with slightly different assumptions are available. These are Wiest's heuristic method (SPAR1)(1967), and Talbot's implicit enumeration method (1982). Some modifications are made in order to apply these methods to the same problem. In section 7.1 and 7.2, outlines of the methods by Wiest and Talbot are given, and modifications made in each of them are identified.

The two modified methods and the intensity assignment algorithms will be used in scheduling some projects and results will be compared. The most satisfying results would be obtained if the project data were real. But we have two limitations in using real project data in comparison. First, it is very hard to find a reasonable number of real projects which will give a fair number of statistics for comparison. Second, large, realistic-size projects are too big to be handled by Talbot's method efficiently. Consequently, arbitrarily generated small projects will be used in comparison of methods. In section 7.3 , the explanation of the arbitrary network generation is given.

Four methods, Upgrading-Only, Upgrading and Downgrading, Wiest's heuristic, and Talbot's Implicit Enumeration, are used to schedule 120 arbitrarily generated projects. Results are presented in section 7.3. Comparison of results are made. Note that, the methods by Wiest and Talbot are not originally designed to solve the problem with our assumptions, so , the comparison of results is only aimed to provide evidence of satisfactory performances of the intensity assignment algorithms, not to find the dominating method.

7.1. Review of Methods Used in Comparison

7.1.1. Wiest's Heuristic Model

Wiest's heuristic model focuses on available resources like intensity algorithms do. It serially allocates, period by period, to activities available to start. Activities are scheduled, starting with the first period, by selecting from the list of those currently available and ordered according to their total slack. The activity with the least slack has the highest priority of being scheduled as available resources permit.(Note that, SPAR1 uses a probabilistic dispatching rule, in which the least slack activity has the highest probability of being scheduled. A simple priority dispatching rule is used here since a one pass solution will be used. A multi pass solution was originally used in SPAR1 to cope with minimizing the total cost of the schedule.) If an available activity fails to be scheduled in that period, an attempt is made to schedule it in the next period. Eventually, activities postponed become critical and move to the top of the priority list of available activities.

The basic model described above is modified by a number of additional scheduling heuristics which are designed to increase the use of available resources and/or to decrease the length of the schedule.

Crew Size: With each activity is associated a normal crew size, ie, the number of men normally assigned to the activity; a maximum crew size, or the maximum number of men which can be assigned to the activity; and a minimum crew size, or the smallest number of men which can be assigned to the activity. Note that, these crew sizes correspond to unit intensity, intensity upper bound, and intensity lower bound , respectively, in the variable intensity model, and they will be used accordingly in solutions of arbitrary problems. The rules for crew size selections are as follows: If an activity to be scheduled is critical, it is placed on a priority list and given special treatment. If sufficient men are available, the activity is scheduled at its maximum crew size. If insufficient men are available to do so, or even to schedule the activity at normal crew size, then an attempt is made to obtain the required men by means of 'Borrow and Reschedule' routines described below. If all efforts fail, and the activity cannot be scheduled at even minimum crew size then early start date is delayed one period. Non-critical activities are scheduled at normal crew size if the required number of men are available, but if the resources available are insufficient for scheduling even the minimum crew size then the activity is delayed for consideration until the next period. Note that, an activity is considered to be critical if its slack does not exceed a value of k, where k is an input parameter. Zero is assumed for the value of k in the experiments.

Augment Critical Activities: Repeated attempts are made to speed up critical activities which have crew size less than their maximum crew size. Before any new activities are scheduled on a given period, activities previously scheduled and still active are examined. If any of these is critical and has a crew size assigned less than its maximum, and if resources are available, its crew size is increased up to the maximum.

Multi-Resource Activities: Separate activities are created for each resource and activities are considered to start at the same period and with the same level of resource assignment. Note that the variable intensity model handles multi-resource activities without creating any extra activities.

Borrow From Active Activities: If resources available are insufficient for scheduling some critical activity j, then the model enters into a procedure for searching currently active activities to see if sufficient men might be borrowed from them to schedule the activity j. Men are borrowed from activities only when the resultant stretching of the activities will not delay the entire project.

Reschedule Active Activities: If sufficient men could not be obtained from the borrow routine described earlier, then the model scans the list of currently active activities and picks out those which could be postponed without delaying the project finish time. If sufficient men is are found in this way, then activity j is scheduled and necessary adjustments are made. Step 3 of the intensity assignment algorithms performs a similar procedure.

Add-on Unused Resources: After as many activities as possible are scheduled on a given day, there still may be unused resources in some shops for that day. The model compiles a list of active activities to which these resources might be assigned and arranges them in ascending order of their total slack. Proceeding down the list, the model increases the crew size of these activities until the unused resources are fully allocated or else the list is exhausted. A similar procedure is performed in Step 4 of intensity assignment algorithms.

Wiest's model is designed to find the schedule with minimum cost for the overall project. The model tries to generate many different schedules and each schedule is evaluated by a cost function. After a number of iterations the minimum cost schedule is chosen by a search routine. Since our interest is in minimizing the project duration, search routines will not be used in experiments. A one-pass solution will be used.

Wiest's model basically assumes three alternative levels of resource applications for each activity (Normal crew size, maximum crew size, minimum crew size). In his article, Wiest mentions that some intermediate crew sizes may be selected although it is not shown how it is done in the flow chart presented in the article (Wiest (1967)). He proposes the intermediate crew sizes to still have discrete values for the number of men per day. Our version of Wiest's model uses the variable intensity model which allows a continuous range of resource applications within upper and lower bounds. Our version of Wiest's model assumes that intermediate levels of resource applications (crew size) can be chosen only when augmenting the critical activities.

Wiest's model and the intensity assignment algorithms have similar heuristics for exploiting the use of available resources and decreasing the project length. The orders of application of those heuristics are different in each method. For example, Wiest's model first augments all the critical activities in the active list and then starts scheduling the ready activities. The intensity assignment algorithm puts both ready and active activities into a combined list which is sorted by their slacks and augments from this list. Earlier experiments with the intensity assignment algorithms suggest that using a combined list produces better schedules then augmenting first the active list and then the ready list.

A FORTRAN code was created for Wiest's model. Originally, Wiest used periodic review for resource assignments, but event-based review is used in this code which is more efficient. Resource assignments are made only when there is a change in the system. Continuous resource application is assumed.

7.1.2. Talbot's Implicit Enumeration Method

Talbot (1982) discusses a model which is more general than the classical resource constrained project scheduling problem. The model permits each activity to be accomplished in one of several different modes. Each mode has predefined resource requirements and duration. A schedule specifies how each activity should be accomplished (in which mode), that is, which resource-duration option should be selected, and when each activity should start. A two phase implicit enumeration methodology is proposed to solve the problem. The procedure proposed gives optimal solutions for small size problems but works as a good heuristic for larger problems.

Talbot's model assumes resources consumed and durations of activities are discrete. Once a mode is selected for an activity, resources used by the activity are constantly applied from its start to its finish. This is quite different than the assumptions of the variable intensity model, namely, that intensity of an activity may vary between its bounds during the execution of the activity.

In order to apply Talbot's model to the same problem with intensity assignment algorithms, some assumptions are made. It is assumed that all modes of an activity have the same resource requirement mix but application rates of each mode are different. Each activity has a number of modes of operation (K_i). The number of modes of activity i is calculated as follows:

$$K_i = \frac{\bar{x}_i - \underline{x}_i}{0.1/d_i} + 1$$

Intensity upper bound of the activity corresponds to fastest mode (mode 1) and intensity lower bound of the activity corresponds slowest mode (mode K_i). An intermediate mode k represents an intensity level x_{i_k} in the variable intensity model, which is:

$$x_{i_k} = \bar{x}_i - (k-1)\,0.1/d_i \tag{7.1}$$

Note that this modification of Talbot's model still has discrete alternatives for resource applications. So the optimal solution for this case may not be the optimal solution for continuous variable intensity case.

Solution Methodology

Talbot developed a two-stage methodology. Stage one defines the problem as a compact integer programming problem, while stage two searches for the optimal solution using an implicit enumeration scheme that systematically improves generated heuristic solutions.

Stage One

In stage one the network is first relabeled using one of the scheduling rules. Talbot presents results for eight different scheduling rules in his paper, and the Minimum Late Finish rule time gave the best performance for the minimum project duration problem. The fastest mode for all activities is assumed in the late finish time calculations. In our experiments, the Minimum Late Finish rule is used to relabel the network. The effect of this relabeling procedure is to specify the order in which activities will be considered for assignment during stage two of the algorithm. Resources and modes are also sorted in stage one. Resources are sorted such that the resource having the maximum frequency of highest per period requirement relative to average availability has the smallest numerical label. The effect of this sort during the enumeration procedure is to identify resource infeasibility as early as possible. Modes are sorted by increasing duration. In the experiments, each activity A_i has K_i modes in which m_1 corresponds to the fastest mode $(x_{i_1}(t) = \bar{x}_i)$, m_{K_i} corresponds to the slowest mode $(x_{i_{K_i}}(t) = \underline{x}_i)$ with the rest of the modes arranged according to equation 7.1. Note that the sorting of the modes affects the initial project completion time found by the enumeration procedure.

Stage Two

In this stage an implicit enumeration algorithm builds feasible partial solutions into complete schedules by considering activities for assignment in increasing numeric order. For each activity modes are also considered in increasing numerical order, ie, fastest mode to slowest mode. The algorithm begins with an attempt to find the earliest feasible assignment of activities with labels 1, 2, 3, and so on, in order, until an activity is identified that cannot be assigned even with slowest mode without violating precedence or resource constraints or without exceeding a bound such as its late finish time LF_j. (A large value, ie, sum of the durations of all activities with their slowest mode, is used as the late finish time of the last activity A_N for the first trial. The late finish bounds of other activities are calculated by the CPM method backward pass procedure whereby the duration of each activity corresponds to its fastest mode.) When such an activity is identified the algorithm backtracks to the most recently assigned activity and attempts to reschedule it at a later time in order to free resources for use by the unassigned activity. This process is systematically applied to all activities and modes until an improved solution is found or until optimality is verified. If an improved solution is found, bounds (LF_j's) are correspondingly tightened (ie, the amount of difference between LF_N of current trial and the value of improved project duration is subtracted from the current late finish bounds of activities in order to find the late finish bounds of activities that will be used at the next trial.) and the assignment procedure begins again with activity 1. Ultimately optimality (or termination) is verified in one of two ways. Either all possible activities assignments have been explicitly or implicitly evaluated or a solution is found which equals a proven theoretical bound such as the project duration with minimum possible durations for all activities.

A FORTRAN code of Talbot's method was created. In our experiments, the number of trials to find the optimal solution is restricted to five trials. The computer time for each problem is also restricted to 60 seconds of CPU time. If optimality could not be reached by then, the existing solution (ie, best solution so far) is taken as the schedule.

7.2. Generating Arbitrary Project Networks

Data for a reasonable number of project networks for comparison of project scheduling methods is hard to find. Project networks appear in such a wide range of different contexts that no one class of problems can claim to be representative. Thus, it was decided to generate and solve test problems. Test problems were generated by a computer model which generates arbitrary project networks. Project networks were randomly drawn with the following attributes:

1. Number Of Activities(Length)

   Number of activities was randomly chosen between 30 and 50. This size of project network is not representative of the size of projects that intensity assignment algorithms are designed for. Talbot's method can solve up to this size of problem within a reasonable computing time.

2. Width Of Network

   This is the maximum number of arcs found in any vertical cut in the network. Random manipulation of the incidence matrix representing precedence relations of activities is used to randomize the width of the network. The detailed description of the procedure is given in Appendix 3.

3. Complexity Of Network

This is the ratio of the total number of arcs to the total number of nodes in the network. It is randomized between 1.5 to 4.5. The detailed description of the procedure is given in Appendix 3.

4. Activity Durations

Activity durations are randomized between 0 and 10 units of time. The range is chosen small because the efficiency of Talbot's method depends on the time horizon chosen.

5. Resource Use Of Activities

It is assumed that there are five types of non-storable resources. Each activity can use up to three different types of resources. The types, amounts and number of different resources used are randomized for each activity.

6. Tightness Of Resources

Resource capacities were assumed to be constant through each project execution in all experiments for simplicity. Note that the intensity assignment algorithms can handle varying resource capacities. SPAR1 and TIE methods can also be modified to handle varying resource capacities but the examples in the literature for both of these methods use only constant resource capacities. A percentage of the peak capacity level of each resource in the Early Start schedule of a project is used as the capacity of that resource. Peak capacity levels of resources used in the Early Start schedule of the project are calculated under the assumptions of fixed normal intensities for all activities and the resources of all types are unlimited, ie, as it is in CPM. Each resource capacity is found by multiplying its peak utilization level by a random draw from the [60%,80%] interval.

7. Resource Application Flexibility

The project network generated up to this step is used to generate four parametric problems. Resource application flexibility is represented by a parameter. Flexibility ranges [lower intensity bound, upper intensity bound] are assumed to be equal for all activities in each experiment. Each project network is to be scheduled under the following flexibility conditions for activity intensities.

A. $\underline{z}_i = 1.0/d_i$, $\bar{z}_i = 1.0/d_i$ for all i (Inflexible)

B. $\underline{z}_i = 0.8/d_i$, $\bar{z}_i = 1.2/d_i$ for all i

C. $\underline{z}_i = 0.5/d_i$, $\bar{z}_i = 1.5/d_i$ for all i

D. $\underline{z}_i = 0.2/d_i$, $\bar{z}_i = 1.8/d_i$ for all i (Most flexible)

7.3. Experiments with Methods and Analysis of Results.

Thirty arbitrary project networks were generated by the procedure explained in section 7.2. Characteristics of the arbitrary projects are summarized in Table 7.1. First, projects were scheduled by four methods (1. Upgrading-Only, 2. Upgrading and Downgrading, 3. SPAR1, 4. Talbot's Implicit

Problem Number	Number of Activities	Number of Resources	Network Complexity	Resource Tightness	Critical Path Duration
1	36	5	2.54	65	85
2	37	5	2.03	64	61
3	48	5	1.63	77	50
4	47	5	1.72	64	44
5	31	5	3.45	64	81
6	44	5	1.91	64	119
7	41	5	1.57	64	71
8	37	5	2.11	64	82
9	44	5	2.73	60	90
10	38	5	2.55	62	139
11	44	5	3.02	69	90
12	36	5	2.57	69	44
13	48	5	2.29	61	77
14	44	5	1.36	69	51
15	44	5	3.00	69	77
16	33	5	2.73	63	98
17	39	5	2.38	70	101
18	45	5	3.16	77	84
19	46	5	3.72	76	100
20	42	5	4.10	74	107
21	45	5	3.58	77	84
22	40	5	3.01	78	77
23	38	5	3.11	62	100
24	37	5	2.19	74	70
25	41	5	1.49	66	54
26	47	5	2.28	73	79
27	37	5	2.46	74	110
28	30	5	3.63	79	105
29	36	5	1.94	66	49
30	36	5	2.14	66	51
Average	40.3	5	2.55	68.7	81
Range	30 - 48	5	1.36 - 4.10	60 - 79	44 - 139

TABLE 7.1. Project Characteristics of artificially generated projects which are used in the experiments.

Enumeration (TIE)) for fixed intensity case, ie, such that classical resource constrained project scheduling problems are faced. Th. first three methods produced the same schedule for each of the projects, since they use the same priority dispatch rule for activity starts (Least Slack First). The TIE method produced different schedules for some of the projects. Table 7.2 summarizes the comparative results of the first three methods against the TIE method.

Cases	Frequency	Percentage
Number of Projects in which All Methods Produced Equal Schedules	17	57
Number of Projects that TIE Method Produced Shorter Schedule than Others	7	23
Number of Projects that TIE Method Produced Longer Schedule than Others	6	~~25~~ 20
Total	30	100

Table 7.2 Summary of Project Completion Time Results For Fixed Intensity Case [$\underline{g}_i = 1.0/d_i$, $\bar{g}_i = 1.0/d_i$]

For the fixed intensity case, the TIE method finds the optimal schedule unless it is interrupted. Interruption may happen for one of two reasons: either computer time allowed for the problem is exceeded or the number of trials required to shorten the schedule exceeded five. In the experiments, computer time was the cause of interruption for all of the problems interrupted before finding the optimal schedule for this method. Results show that in 23% of the projects, the TIE method could not find the optimal solution before being interrupted. This reflects the inefficiency of the TIE method even in smaller size problems. In the fixed intensity case there is only one mode for each activity so, the problem size is smaller than the problems with variable intensity case.

The first three methods produced equally good or better schedules than the TIE method in 77% of the projects. This reflects the fact that the Minimum

Slack First priority rule gives good results for the classical resource constrained project scheduling problem. This supports the findings of Davis and Paterson (1975) and many others.

Next, the projects in Table 1 were scheduled by the four methods for three resource application flexibility cases in which each A_i has intensity bounds of $[\underline{r}_i=0.8/d_i\ ,\ \bar{r}_i=1.2/d_i]$, $[\underline{r}_i=0.5/d_i\ ,\ \bar{r}_i=1.5/d_i]$ and $[\underline{r}_i=0.2/d_i\ ,\ \bar{r}_i=1.8/d_i]$, respectively. Project completion time results for these cases are reported in Table 3, Table 4, and Table 5, respectively. The tables include the percentages above the best project finish time by each method for each project. The best finish time of a project is the smallest of the project finish times found by the four methods. The TIE method found the optimal solution for some projects. However, recall that the optimal solution found by the TIE method may not be optimal for the continuously variable intensity case. The TIE method only considers a number of discrete values of intensities for the activities, and cannot assign any value between upper and lower intensity bounds. But for those cases that the TIE method found an "optimal" solution, we expect the solution would be near-optimal for the continuous intensity case, since the number of discrete intensities that the TIE algorithm may pick was chosen reasonably large (ie, $K_i=5$ for the mild flexibility case, $K_i=11$ for the more flexible case, and $K_i=16$ for the most flexible case).†

Table 7.6 provides a summary of project completion time results for the three cases of resource application flexibility. Overall results show that the

† That is, the TIE method would be near-optimal for the continuous intensity case when intensity must be held constant from start to finish of each activity.

	Upgrading Only		Upgrading & Downgrading		SPAR1		TIE	
Problem Number	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best
1	73	9	73	5	73	15	67	0
2	57	6	54	0	71	31	55	0
3	86	21	85	16	85	21	38	0
4	57	12	57	12	81	33	33	0
5	77	3	75	0	80	7	75	0
6	102	11	101	13	115	18	92	0
7	62	9	57	0	61	7	53	11
8	71	4	72	7	74	9	63	0
9	73	3	76	0	65	13	78	0
10	120	5	120	5	124	9	119	0
11	35	8	54	6	83	11	79	0
12	39	0	35	0	42	15	39	0
13	79	0	79	0	54	6	52	4
14	83	8	83	3	51	23	40	0
15	75	7	71	1	61	16	70	0
16	95	10	87	0	101	16	97	12
17	93	7	93	7	97	16	84	0
18	71	3	69	0	71	3	72	4
19	32	2	61	0	54	4	100	24
20	93	8	93	8	93	8	85	0
21	63	12	74	0	62	11	74	0
22	67	5	64	0	73	14	64	0
23	90	1	90	1	92	3	89	0
24	59	2	61	5	61	5	53	0
25	50	11	49	9	53	12	45	0
26	72	6	70	3	79	16	65	0
27	109	9	103	3	112	12	100	0
28	33	11	80	1	89	11	79	0
29	86	10	82	5	95	13	40	0
30	62	13	54	0	62	13	57	6
Totals	2201	224	2135	121	2313	403	2106	74
Averages	73	7.5	71	4	77	14	70	3

Table 7.3. Project Completion Times Results for $[\underline{z}_i=0.8/d_i\ ,\ \bar{z}_i=1.2/d_i]$ resource application flexibility.

	Upgrading Only		Upgrading & Downgrading		SPAR1		TIE	
Problem Number	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best
1	[illegible]	2	50	0	70	14	57	4
2	53	13	45	0	53	29	57	27
3	37	6	35	0	45	29	35	0
4	30	7	29	4	40	43	26	0
5	67	3	67	0	73	16	72	0
6	39	11	62	13	101	26	30	0
7	59	6	51	0	53	4	55	0
8	62	9	51	7	64	12	57	0
9	75	19	64	0	77	20	63	6
10	112	8	111	5	117	10	106	0
11	72	4	69	0	79	15	74	7
12	35	11	35	0	41	17	45	22
13	73	4	70	0	79	11	76	11
14	35	0	35	0	38	5	35	0
15	53	3	62	0	74	17	71	13
16	32	5	73	0	56	10	39	14
17	73	7	77	8	73	7	33	0
18	59	4	57	0	53	11	65	11
19	70	0	73	0	73	4	75	9
20	81	4	73	0	82	5	76	0
21	70	3	63	0	79	9	69	2
22	55	5	52	0	54	4	53	2
23	73	3	73	0	51	7	77	4
24	52	4	50	0	59	8	51	2
25	46	15	64	10	51	23	40	0
26	61	5	62	7	79	29	52	0
27	91	2	94	6	93	5	59	0
28	73	11	71	8	73	11	66	0
29	40	11	35	0	38	6	37	3
Totals	1423	203	1836	72	2051	424	1911	171
Averages	64	7	63	2	65	14	[illegible]	6

Table 7.4. Project Completion Times Results for $[\underline{z}_i = 0.5/d_i\ ,\ \bar{z}_i = 1.5/d_i]$ resource application flexibility.

	Upgrading Only		Upgrading & Downgrading		SPAR1		TIS	
Problem Number	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best	Project Completion Time	Percentage Above Best
1	65	10	60	0	71	18	62	3
2	57	8	45	0	60	25	52	16
3	35	21	32	10	44	52	29	0
4	27	17	25	9	30	43	23	0
5	64	2	63	0	75	21	69	10
6	66	19	33	15	57	21	72	0
7	52	3	43	0	53	10	53	10
8	62	17	54	2	65	25	53	9
9	55	10	59	0	63	15	65	7
10	103	7	106	5	109	3	101	0
11	63	5	65	0	70	3	75	15
12	35	3	35	6	39	12	34	0
13	69	11	62	0	67	0	59	11
14	33	18	32	10	38	31	29	0
15	63	3	61	0	71	16	65	7
16	77	4	74	0	73	5	63	12
17	78	1	73	0	75	4	73	0
18	55	10	50	0	60	20	50	0
19	53	0	63	0	74	9	75	10
20	73	9	70	5	77	15	57	0
21	55	19	54	0	53	7	55	2
22	52	10	50	5	52	10	47	0
23	78	8	43	3	75	6	71	0
24	49	11	44	0	52	18	51	16
25	40	25	25	10	41	23	32	[illegible]
26	62	11	55	0	55	16	57	2
27	63	1	57	0	37	6	37	5
28	65	22	61	13	65	22	54	0
29	30	22	32	0	42	31	32	0
30	52	2	51	0	51	0	57	12
Totals	1,699	302	1725	94	1924	510	1741	159
Averages	60	10	57	3	64	17	58	5

Table 7.5. Project Completion Times Results for $[\underline{x}_1=0.2/d_1 \, , \, \bar{x}_1=1.8/d_1]$ resource application flexibility.

Upgrading & Downgrading algorithm performed best by giving the best project completion time result 51 times out of 90, averaging 3% above the best project completion time, and producing a maximum of 16% above the best completion time. The TIE method performed second best, by giving the best or equally best results 47 times, averaging 4.7% above best, and producing a maximum of 27% above best. Third and fourth best are Upgrading-Only and SPAR1, respectively.

For the less flexible case ($\underline{g}_i=0.8/d_i$, $\bar{g}_i=1.2/d_i$) the TIE method performed a little better than the Upgrading and Downgrading algorithm. For this case, the number of modes in TIE is only five and the method could find an optimum more frequently than in the cases with more flexibility, since less computer time is required.

For higher flexibility cases [$\underline{g}_i=0.2/d_i$, $\bar{g}_i=1.8/d_i$] and [$\underline{g}_i=0.5/d_i$, $\bar{g}_i=1.5/d_i$], the Upgrading and Downgrading algorithm performed better than the TIE method in all measures. Comparing Tables 7.3, 7.4 and 7.5 indicates that increases in resource application flexibility improves the performance of the Upgrading and Downgrading algorithm.

The Upgrading-Only Algorithm's performance was fairly good considering that there is a restriction of management such that once resources are assigned to an activity they can not be withdrawn. In general, the Upgrading-Only Algorithm performed better than SPAR1. Both Upgrading-Only and SPAR1 become noticeably inferior to the other two methods when one increases the flexibility of resource application. Both of these methods are more prone to fall into the case that resources are tied up unnecessarily by some nonurgent

Resource Application Flexibility	Average Percentage Above Best Solution				Number of Problems Methods Performed Best Including Ties				Maximum Percentage Above Best Solution			
$\underline{z}_i$, $\overline{z}_i$	1	2	3	4	1	2	3	4	1	2	3	4
$0.8/d_i, 1.2/d_i$	8	4	14	3	1	12	0	21	21	16	33	24
$0.5/d_i, 1.5/d_i$	7	2	14	6	3	·21	0	11	19	13	43	27
$0.2/d_i, 1.8/d_i$	10	3	17	5	1	18	1	15	25	13	52	16
All Cases	8.3	3.0	15.0	4.7	5	51	1	47	25	16	52	27

Table 7.6 Summary of project completion time results for three cases of resource application flexibility

activities than is the Upgrading and Downgrading algorithm.

Table 7.7 summarizes the relative performance of the methods with respect to project finish time results of the 120 experiments (including the inflexible resource application case). Results again indicate that the Upgrading and Downgrading Algorithm performed the best of all, with 74 shortest durations, 40 of which were the unique shortest duration.

	METHODS			
Cases	Upgrading Only	Upgrading & Downgrading	SPAR1	TIE
Number of Times Shortest Schedules Produced Including Ties	28	74	24	72
Number of Times Unique Shortest Schedule is Produced	0	40	0	46
Number of Times Longest Schedule is Produced	3	0	62	18

Table 7.7 Relative Performance of methods to Project Completion Times results for all experiments including inflexible resource application case.

In none of the test problems did the algorithm provide the unique longest duration for the project. Talbot's method gave the shortest duration 72 times, 46 of which were the unique shortest duration, and the method produced the unique longest duration 18 times. The Upgrading-Only Algorithm found the shortest durations 28 times including ties. In none of the cases was the shortest duration unique, and in only three cases did it produce the unique longest project duration. With respect to the number of longest durations found by the methods, Upgrading-Only performed better then TIE method by 3 to 18.

7.4. Computational Experience

Projects were scheduled by each of the four methods with an IBM 3081 computer. Each project network in Table 7.1 was solved for the four different resource application flexibilities at once in order to reduce the cost of the whole experiment. The recorded computation times were for the batch of four problems by each method. Table 7.8 summarizes the computing time results for the batches of four problems. Talbot's method finished the batch of four problems within the allowed computer time 18 times out of 30. Allowed time for a batch of four problems was chosen to be 200 seconds of CPU time. In the cases that the 200 second time limit was exceeded, individual runs were made with a 60 second time limit of those problems which were not finished in the batch. In this way, a time limit of 60 seconds for each problem was allowed. The results reported below for Talbot's method exclude the cases that the program exceeded the allowed time limits.

	METHODS			
Cases	Upgrading Only	Upgrading & Downgrading	SPAR1	TIE *
Range of Computing Times (in seconds)	1.31 - 2.57	2.09 - 8.91	1.20 - 2.11	0.45 - 134.8
Average of Computing Time (in seconds)	2.29	4.27	1.55	26.25

Table 7.8 Summary of Computation Time Results on an IBM 3081.(All times are CPU times.)

* Statistics given for this method are calculated by excluding the cases that the program could not find the solution within allowed computer time.

The first three methods have stable ranges of computing times and reasonable average computing times. SPAR1 has the lowest average of 1.55 seconds. Its highest computing time was only 2.11 seconds for a batch of four problems. Upgrading-Only and Upgrading and Downgrading have average computing times of 2.29 and 4.27, and maximum computing times of 2.56 and 8.91 for a batch of four problems, respectively. These results indicate that none of the first three methods would be seriously limited by computing time when scheduling larger projects, ie, projects with several thousand activities and hundreds of resources.

On the contrary, the computing time for Talbot's method varies drastically for different problems. Even though the project networks were chosen to be of small size, the program could not finish twelve out of thirty cases. Excluding these cases, the average computing time was 26.25 seconds and the maximum computing time was 134.6 seconds for the batch of four problems. Experience indicates that computing time in this method is affected by problem size, network complexity and resource application flexibility (number of modes for each activity).

The computing time results suggests that Talbot's method is too inefficient to solve (in a practical manner) problems much larger than the problem size we chose in experiments. Yet, the size of many real projects is several multiples of the size of the problems picked in the experiments.

7.5. Conclusion

The analysis of results has shown that the Upgrading and Downgrading method produced better results for project completion times compared to the others. Results for the Upgrading-Only Algorithm are fairly good considering that an additional restriction is applied by management. Results also indicate that both algorithms are computationally efficient for scheduling large projects. The TIE method produces near optimal schedules unless it is interrupted. Computational experience suggests that Talbot's method is too inefficient to handle problems of larger size real project networks.

Both SPAR1 and TIE methods were not originally designed for handle variable intensities for the activities. These methods were modified for the purposes of comparison of results of these two methods to the results of intensity assignment algorithms. The validity of the results that can be obtained from these algorithms is evident.

8. HANDLING STOCHASTIC ELEMENTS IN PROJECT NETWORKS

8.1. The Approach for Handling Stochastic Elements

There are two basic types of uncertainty in project-oriented production networks. The first type is the uncertainty as to the actual amount of man-hours required to complete a production activity. The second type is the uncertainty as to the nature and amount of repair rework that may be needed after a *test activity*, ie, one is faced with a stochastic project network.

We shall use Monte Carlo simulation to treat the stochastic elements. The reason for simulating the project schedules is because stochastic elements in the project cause some of the important parameters of the project schedules to be random variables, such as, important milestone dates, project completion date, and total use of resources in certain periods. The probability distributions of such random variables cannot be analytically calculated from parameters for the activities of the project, because of the complex convolutions involved in any large network. By repeated simulation of randomly selected values for the stochastic elements in the network, approximate probability distributions for important schedule parameters can be developed.

The methods developed to handle stochastic elements for project scheduling in the literature do not provide treatments directly applicable to the stochastic elements discussed in chapter 4. Consequently, a simulation model is developed which handles stochastic resource requirements for activities, handles stochastic network structure of the project and uses the intensity assignment algorithms to exploit the flexibility of resource application to the activities which is available.

The approach requires modeling the structure of each stochastic element. Models for resource requirements of production activities and test activities will be presented. In the approach, a number of different schedules are generated by using random values for the parameters of stochastic elements in each simulation. One of the intensity assignment algorithms will be used as a decision maker for resource allocation among activities. The resulting schedules from simulations are used to developed approximate, unbiased distributions for important milestones of the project and for total use of different types of resources during the project execution.

6.2. Stochastic Resource Requirements

We suppose there are two different kinds of activities classified according to their stochastic nature. Those in the first type are called "production activities" which include manufacturing, repairing and rebuilding types of jobs. Those in the second type are called "test activities" which involve the performance of tests on various systems. For production activities, total resource requirements are stochastic. PERT methods attempt to model this by assuming that activity durations are stochastic. In the variable intensity model, the activity duration is a function of resource applications. Stochastic resource requirements provides a better representation of work uncertainty in shipyards than stochastic activity durations.

Total resources required by an activity are assumed to have distributions with known parameters. This is the case for the projects of a shipyard. The shipyard has past history of total man-hours from each resource required by activities. Distributions can be driven from the histories.

8.3. Stochastic Project Network

Resource requirements of test activities are fairly stable for performing the activity itself, but every test activity may result in a success or a failure. If it is a success then its regular followers may start, but if it is a failure , then there needs to be some kind of repair rework utilizing different resources than those used by the test. The repair needed may change with the type of failure in the particular activity. There are a number of possible repair procedures (activity subnetworks) which may follow each test activity. After one of the repair procedures is completed, the test activity is executed again with discounted parameters for the chance of further rework. Before project execution, it is uncertain whether a test activity will result in a success or a failure (or even repated failures). If it is failure, the repair subnetwork to be undertaken is uncertain, so, the network of the project is stochastic.

Model For Test Activities

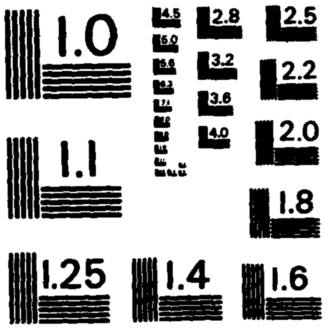
Assume A_i is a test activity. Once its predecessors are finished A_i may start and perform the test using resources (a_{ki}). The result of A_i can be either success or failure. If it is success its regular (deterministic) followers may start†, but if it is a failure, one of the repair subnetworks (R_i) must be urdertaken. After a repair subnetwork is completed (A_i) is to be performed again. A network

† Strict precedence is assumed for the followers of test activities.

AD-A159 980 PROJECT SCHEDULING IN PROJECT-ORIENTED PRODUCTION SYSTEMS(U) CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER A DINCERLER SEP 85 ORC-85-11 2/2

UNCLASSIFIED F/G 13/8 NL

END
FILMED
DTIC


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

representation of a test A_i and follow-on rework is given in Figure 8.1. In addition to previously defined variables, test activities have the following parameters;

p_i probability that test A_i is a success at the first trial.

α_i increase in probability of success after each failure of test A_i. $(0<\alpha_i<1)$

γ_i discounting factor for resource requirements after each failure of test A_i. $(0<\gamma_i<1)$

R_i $= \left\{ r_1 , r_2 , \ldots , r_{m_i} \right\}$

set of possible ***repair routes*** for test A_i. Note that, a route is a subnetwork of activities. For each activity in any route, resource requirements, duration, and precedence relations must be specified. Generally, activities in repair routes require different mixes of resources than the test activity requires.

dr_{il} normal duration of repair subnetwork r_l after test A_i.

q_{il} probability that repair subnetwork r_l is executed given a failure of test A_i. $\sum_l q_{il} = 1.0$

The probability that test A_i is success (p_i) is increased by a constant factor $(1+\alpha_i)$ after each failure. The resource requirements of A_i are discounted by a factor (γ_i) after each failure. Given a failure of test A_i, the probability that a repair subnetwork is to be executed (q_{il}), is assumed to be trial-invariant.

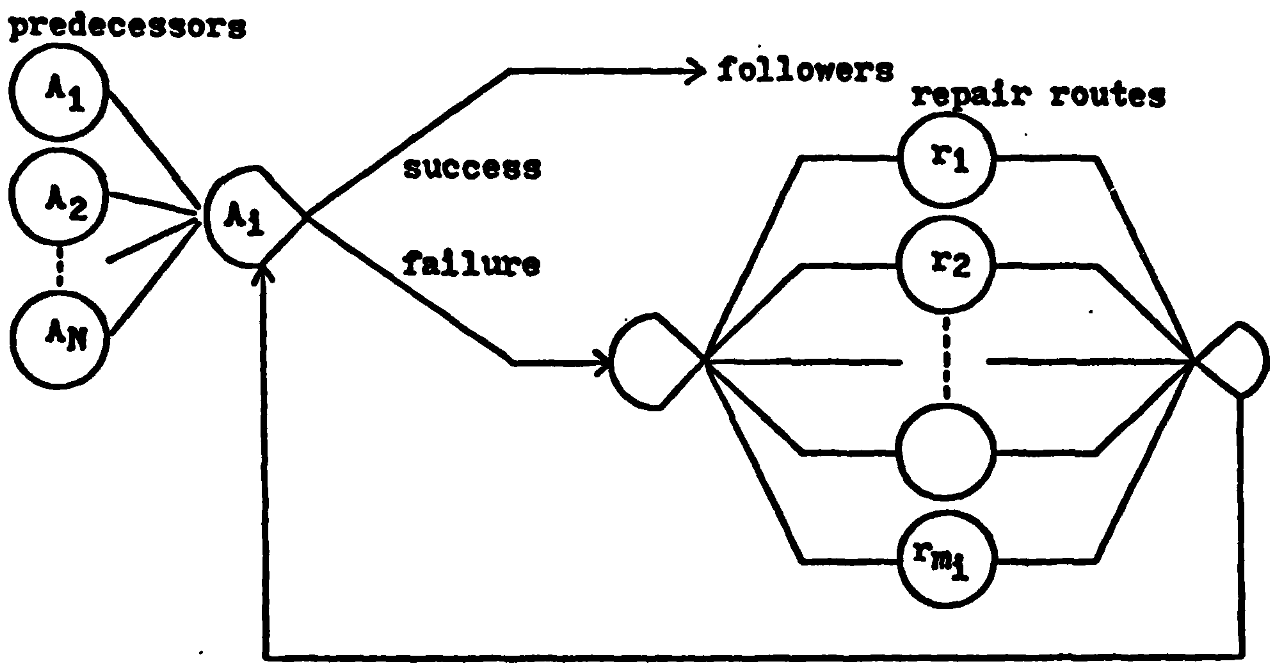


Figure 8.1 Network representation of test activities.

5.4. Treatments for Stochastic Elements in Simulations

At the beginning of each simulation of the project, resource requirements of all production activities are randomized. That is,

$a_{ki} \leftarrow a_{ki}\ random_i$

where $random_i$ is a random number drawn from a known multiplicative distribution for A_i with mean 1.. Then, randomized resource requirements are used as actualized resource requirements for the simulation.

Test activities are treated differently at their finish times. After each test the following procedure is performed.

1. Determine if the test is a success or a failure, by a random draw.
2. If a failure, determine the repair route to be taken, by a random draw.
3. Do updates for next trial of test A_i.
   a. Discount resource requirements:

      $a_{ki} \leftarrow a_{ki}\ \gamma_i$ for all k

   b. Increase the probability of success:

      $p_i \leftarrow p_i + \alpha_i$

   c. Update following parameters for the test A_i.

      $F_i(t) \leftarrow T$ where T is a big number distinguishing that test A_i must be repeated.

      $s_i(t) = 0.$ and $pcv_i(t) = 0.$

Since the probability of success improves after each failure, in a finite number of trials test i will be a success and its followers may start.

Considering the durations of rework activities which follow the test activity, the overall duration of testing and rework can fluctuate greatly. This raises the question of what values for test durations to be used in (target) late finish time calculations which are used by intensity assignment algorithms. We have two choices for test durations. One is , assuming that tests will not fail, take the normal duration of a test as its duration. Second is, find expected values using its parameters. The first approach can be criticized to be myopic. Since there is a substantial risk that the overall testing duration may be multiplied, we might choose misleading target times associated with the wrong critical path. As for the second approach, using expected durations in CPM calculations has been shown to produce overly optimistic schedules in the literature (for fixed intensity case) (Clark (1962), MacCrimmon and Ryavec (1964)). On the other hand, allowing variable intensities for activities gives us more correction possibilities than fixed intensity models. Expected durations for test activities (including rework) are used in late finish time calculations.

8.4.1. Expected Durations For Test Activities

The expected duration for a test activity i is calculated as follows:

$$ed_i = d_i + (E[N_i] - 1) (d_i + erd_i)$$

where,

$E[N_i]$ Expected number of trials until success for test A_i. After every failure the value of this term is updated, since probability of success increases. Initially,

$$E[N_i]=p_i+2(1-p_i)(p_i+\alpha_i)+3(1-p_i)(1-(p_i+\alpha_i))(p_i+2\alpha_i)+....+$$

$$..+L_i(1-p_i)(1-(p_i+a_i))...(1-(p_i+(L_i-2)a_i))(p_i+(L_i-1)a_i)$$

where L_i is the maximum number of unsuccessful trials that test A_i may have, ie. $(L_i=[(1-p_i)/a_i]+1)$†. After k unsuccessful tests,

$$E[N_i]=(p_i+k a_i)+2(1-(p_i+k a_i))(p_i+(k+1)a_i)+...+$$
$$(L_i-k)(1-(p_i+k a_i)...(1-(p_i+(L_i-2)a_i)(p_i+(L_i-1)a_i)$$

erd_i Expected repair duration following an unsuccessful test A_i. $(=\sum_{l=1}^{m_i} q_{il}\, dr_{il})$

Note that expected durations of test activities are recalculated after each failure and the most recent values are used as activity durations in step 2 and step 3 of the intensity assignment algorithms.

6.4.2. Expected Late Finish Updates

Recalculations of expected late finish times of activities are made before every assignment of intensities in the simulations. The procedure described for late finish updates in section 6.1.3 can easily be modified for expected late finish updates. We need to change the step 1(b) of the procedure in section 6.1.3. Step 1(b) will be changed to following expression.

(b) If the activity is in progress, then

$$AF_i(t-1) = t-1 + ed_i - pcv_i(t-1)\, d_i$$

Note: for a production activity i $ed_i=d_i$, so that for production activities,

$$AF_i(t-1) = t-1 + (\, 1 - pcv_i(t-1)\,)\, d_i$$

We also need to substitute ed_i for d_i is step2 of the procedure in section 6.1.3.

† We assume a_i is chosen so that L_i is integer.

9. SCHEDULING A SHIP OVERHAUL PROJECT

The model developed for project scheduling in project-oriented production systems was used to schedule a ship overhaul project. Data of the project was provided by a naval shipyard. The description of the project is given in section 9.1. Several experimental schedules were generated by using intensity assignment algorithms under different assumptions discussed in section 9.2.

First, 'deterministic' schedules were generated for two different resource application flexibility conditions. The deterministic schedules assume that the resource requirements of the activities are deterministic, and also that the project network is deterministic . Then, stochastic simulations were made using the Upgrading-Only algorithm as the decision maker for the resource applications among activities (Section 9.2). Then, the results of experimental schedules are compared and their implications for the shipyard management are discussed in section 9.3.

9.1. Ship Overhaul Project

9.1.1. Project Activities

The project network describing the ship overhaul consists of 2931 activities. They are classified as:

(i) Those requiring resources with deterministic followers, ie, production activities (Type 1)

(ii) Those requiring resources with stochastic followers, ie, test activities (Type 2)

(iii) Those requiring no resources and that have fixed duration, ie, time restraints (Type 3)

(iv) Dummy activities for precedence relations, ie, those that have zero durations (Type 4)

The number of activities of each type were:

Type 1	2027
Type 2	249
Type 3	198
Type 4	457

Activities of Type 3 were required to establish the consistency between the due dates specified by the aggregate planning level and the activities of the project network. The number of activities in the deterministic network were 2465, and the number of repair rework activities were 466.

The average number of successors to an activity was 2.36, indicating substantial connectedness of the activities in the network. The number of overlapping precedence relations (ie, flow transfers) between activities was 536, which was eight percent of the total precedence relations.

9.1.2. Resources, Activity Durations, Resource Requirements

The service of 38 different shops are used to carry out the ship overhaul. Each shop has a number of work centers which correspond to a specific type of trade labor, ie, a nonstorable resource type. Table 9.1 tabulates a list of important shops. The total number of shop-work center pairs, ie, the number of resource types, was 193. Each activity may require services from a number of

SHOP NO.	SHOP NAME
13	Quality Assurance- Engineering and Analysis
33	Quality Assurance- Non-nuclear Inspection
34	Quality Assurance- Metallurgy and Material Control
33	Quality Assurance- Non-destructive Testing
30	Production Department
11	Structural Group- Shipfitters
17	Structural Group- Sheet Metal
26	Structural Group- Welders
41	Structural Group- Boilermakers
81	Structural Group- Foundry
06	Mechanical Group- Tool Shop
23	Mechanical Group- Forge
31	Mechanical Group- Inside Machinist
38	Mechanical Group- Machinist Marine
56	Mechanical Group- Pipefitter
51	Electrical Group- Electrical
67	Electrical Group- Electronics
64	Service Group- Woodmakers
71	Service Group- Painters
72	Service Group- Riggers
99	Temporary Service
32	Nuclear Engineering

Table 9.1. The list of important shops that the overhaul project needs service for execution.

resources. The maximum number of resources required by an activity was 70, and the average number of different resources required by an activity was 9.7.

An activity may contain a number of elementary work packages. For each work package of an activity, a time duration is estimated. A work package is called a Key-Operation. The estimated duration of an activity is the maximum of the durations of its Key-Operations. In the case of Dummy activities, the estimated duration is zero, and for activities of Type 3, activity durations correspond to fixed time lags.

Total shop service hours required for an activity are estimated by summing over the shop service hours required by the Key-Operations of the activity. The shop service hours for activities of Type 3 and Type 4 are zero.

The size of the ship overhaul project may be appreciated by adding the service man-days over all activities of each shop as illustrated in Table 9.2.

9.1.3. Bounds On Activity Intensities

In a practical case when data is available concerning work space and structure being worked on, minimum and maximum shop hours per unit time can be estimated directly for each activity. However, for this study the data on practical limits were not available, and bounds on the activity intensities had to be determined from suppositions.

A uniform upper bound of $1.2/d_i$ was taken for activity intensities, ie, a daily allocation of shop hours twenty percent greater than needed to complete the work in the estimated "normal" time (d_i) would not be exceeded. A uniform lower bound of $0.5/d_i$ was taken for the activity intensities, ie, an activity can not be extended more than 200 percent of its original duration. In this way a daily allocation of shop hours can not be less than fifty percent of the shop hours needed to complete the work in the estimated "normal" time. These bounds apply only to Type 1 and Type 2 activities, as Type 3 and Type 4 activities have fixed intensities, ie, upper and lower bounds are equal to $1/d_i$.

9.1.4. Limitations on Resources

The aggregate planning level provided man-hour availabilities of each type of resource (Shop-Work center), ie, preallocated shop capacities for the project. Man-hour availabilities are given as the daily man-hour availabilities from each shop-work center pair. The daily availabilities change at the beginning of each month (~20 working days).

9.1.5. Milestones

The events that are important in project executions are called milestones.

SHOP	MAN-DAYS
2	2526
3	181
5	0
6	1617
7	26
9	268
10	57
11	12288
12	0
13	161
14	4
16	0
17	9049
19	2875
20	58
22	179
23	246
24	560
26	12723
30	1635
31	14294
32	1262
33	348
34	2751
35	2562
36	2
37	2
38	25248
39	2235
40	0
41	6240
42	20
47	1
50	0
51	19867
56	40735
57	15036
62	1257
64	5544
67	12780
70	31
71	12611
72	17189
81	343
94	101
99	11645
TOTAL	236562

Table 9.2 Total estimated man-days service needed to overhaul the ship. Note that the figures do not include the repair rework requirements.

Ten milestones of interest in the ship overhaul are as follows:

(1) 318 Start steaming Plant 2

(2) 321 Start steaming Plant 1

(3) 825 Start Cold Operations in Plant 2

(4) 828 Complete Cold Operations in Plant 2

(5) 833 Complete Hot Operations in Plant 2

(6) 835 Start Cold Operations in Plant 1

(7) 838 Complete Cold Operations in Plant 1

(8) 843 Complete Hot Operations in Plant 1

(9) 918 Production Work Complete

(10) 999 All Services Complete

9.2. Deterministic Experiments

Four deterministic schedules were generated for the conditions given below. Deterministic schedules were generated by assuming the resource requirements of activities are certain and equal to the expected man hour service requirement from each work center, and by assuming the test activities will be successes at their first trials, ie, no rework activities are actualized. Schedules were generated under the following conditions:

(1) Resources are unconstrained , ie, unlimited resource capacity. Resource applications are inflexible, ie, activities have fixed durations ($\underline{g}_i = \bar{g}_i = 1/d_i$). Note that CPM assumes these conditions; the schedule generated here is equivalent to the Early Start Schedule in the CPM method.

(2) Resources are constrained according to aggregate plans, and resource applications are inflexible for activities. Note that the classical resource constrained problem assumes these conditions (although most solution

procedures for the classical problem cannot cope with time-varying capacities). Both intensity assignment algorithms yield the same schedule in this case since they use same priority rule.

(3) Resources are constrained and resource applications are flexible, ie, $\underline{\varepsilon}_i = 0.5/d_i$ and $\bar{\varepsilon}_i = 1.2/d_i$. For these conditions the project is scheduled differently by the two intensity assignment algorithms; Upgrading-Only Algorithm (3A) and Upgrading and Downgrading Algorithm (3B).

Results of actualization of important milestone dates are summarized in Table 9.3

Milestones	Unconstrained Fixed Intensity (1)	Constrained Fixed Intensity (2)	Constrained Variable Intensity Upgrading Only (3A)	Constrained Variable Intensity Upgrading & Downgrading (3B)
318	265	455	315	293
321	265	486	322	305
825	277	450	352	334
828	319	460	377	360
833	355	355	404	398
835	291	450	326	329
838	335	475	350	352
843	371	371	388	393
918	305	530	331	332
999	525	553	531	529

Table 9.3. The summary of results for milestone actualization times for deterministic experiments.

Comparison of schedule 1 and schedule 2 indicates that there are some tight resources in schedule 2 which cause the delay in actualizations of all milestones with respect to actualization dates obtained from schedule 1 in which resources are unconstrained.

Comparison of schedule 2 with schedule 3A and 3B indicates that flexibility in resource applications improves the schedule for the milestone actualization dates generated by both intensity assignment algorithms. When resources can be applied in variable rates, scheduling under the assumption of fixed durations and fixed resource application rates inhibits productivity. Comparison of schedule 3A with schedule 3B indicates that having a managerial restriction over resource applications, ie, resources can not be withdrawn from an activity once they are committed until the activity finishes, also causes increases in milestone dates.

9.3. Stochastic Simulations

To perform stachastic analysis, total man hour requirements of each activity from each work center was assumed to be a random draw from a uniform distribution between 80% and 120% of its estimated value, since the distributions based on historical man hour requirements were not available for this study.

Repair rework subnetworks following 249 test activities were developed by the shipyard. Estimated probabilities, resource requirements , and durations for repair activities were also provided.

The Upgrading-Only intensity assignment algorithm was used as a decision maker for resource allocation between overhaul activities in all simulations. The Upgrading-Only algorithm was chosen because current decision making for allocation of resources between activities in the shipyard can be best represented by this policy.

40 different schedules were generated by simulations. At each simulation, man hour service requirements were randomized, and a different subnetwork after each test activity was actualized due to randomizations. The statistics calculated considering all simulations give distributions for some important project parameters. Table 9.4 gives the cumulative probability distributions of actualizations of milestone dates. Table 9.5 provides distributions of man hour usage histories through project life for an example shop (Shop 38). Table 9.6

provides the distributions for total man-hours consumed by all project activities.

9.4. Conclusion

The deterministic experiments have shown that using flexibility in resource applications improves the schedule for milestone actualization dates generated by both algorithms. The shipyard actually applies flexibility in resource applications to the project activities. Methods that assume fixed resource application rates for activities may give overly conservative schedules for overhaul projects of the shipyard. Thus, shipyard management should apply a schedule which is generated by a method that assumes flexible resource applications for project activities as its guidelines for scheduling the activities and projecting the schedule parameters.

The deterministic experiments have also shown that increase in flexibility in resource applications to the project activities improves the schedule for milestone actualization dates. In actual practice, the shipyard management generally does not allow withdrawal of resources from an activity before it is finished although it allows supplementing the resources to speed up the activity execution. This is the policy which is used in Upgrading-Only Algorithm. This policy has advantages for the shop managers, ie, it avoids complexity concerning labor reassignments. The disadvantage of this policy is that it may result in longer durations for projects. The shipyard management should consider the resource assignment policy in order to decide if the policy is beneficial to the shipyard. The benefit of finishing the projects earlier may be worth tolerating more managerial difficulties at the shop level. The schedule generated by the Upgrading and Downgrading algorithm has shown that the policy change will reduce the milestone actualization dates.

CUMULATIVE PROBABILITIES

		10%	20%	30%	40%	50%	60%	70%	80%	90%	100%				
MILESTONE	318	297	298	301	304	305	314	317	318	321	325	MEAN=	309.	STD DEV=	9.
MILESTONE	321	309	311	313	315	317	326	327	330	335	342	MEAN=	321.	STD DEV=	10.
MILESTONE	825	333	335	338	341	342	350	353	355	358	362	MEAN=	346.	STD DEV=	9.
MILESTONE	828	357	360	364	369	372	378	380	383	387	392	MEAN=	373.	STD DEV=	11.
MILESTONE	833	399	405	406	410	415	420	421	428	430	432	MEAN=	415.	STD DEV=	11.
MILESTONE	835	327	345	346	350	353	354	356	357	361	364	MEAN=	350.	STD DEV=	11.
MILESTONE	838	350	371	372	377	379	380	383	385	387	391	MEAN=	376.	STD DEV=	12.
MILESTONE	843	391	411	414	417	421	422	425	427	429	435	MEAN=	417.	STD DEV=	12.
MILESTONE	913	331	334	335	336	336	339	341	341	342	345	MEAN=	337.	STD DEV=	5.
MILESTONE	9999	545	551	554	557	560	566	574	582	585	599	MEAN=	564.	STD DEV=	15.

Table 9.4. Cumulative Probability distributions of milestone actualization times.

CUMULATIVE PROBABILITIES

LAST WORKING-DAY OF PERIOD	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%				
37	47	49	50	50	50	51	52	53	55	58	MEAN=	51.	STD DEV=	3.
58	40	43	44	44	45	45	45	47	49	49	MEAN=	45.	STD DEV=	2.
79	42	46	54	55	57	57	59	63	109	134	MEAN=	64.	STD DEV=	25.
98	231	244	246	249	253	255	260	261	275	291	MEAN=	254.	STD DEV=	15.
119	520	538	545	581	595	607	618	624	629	645	MEAN=	501.	STD DEV=	50.
140	764	805	810	828	842	866	893	900	958	1037	MEAN=	856.	STD DEV=	60.
161	951	1013	1051	1081	1102	1120	1144	1181	1240	1330	MEAN=	1106.	STD DEV=	97.
182	1207	1297	1349	1447	1476	1509	1528	1567	1591	1658	MEAN=	1437.	STD DEV=	130.
203	1224	1294	1319	1377	1384	1401	1440	1473	1498	1577	MEAN=	1382.	STD DEV=	91.
225	693	757	789	813	829	839	804	919	967	1111	MEAN=	841.	STD DEV=	96.
246	550	625	653	682	703	713	732	738	749	803	MEAN=	682.	STD DEV=	71.
267	379	407	410	417	423	450	453	456	476	494	MEAN=	430.	STD DEV=	35.
288	248	275	281	285	296	332	344	353	380	391	MEAN=	312.	STD DEV=	44.
309	137	144	153	163	237	436	502	582	615	750	MEAN=	342.	STD DEV=	202.
330	449	569	593	635	640	694	716	724	755	862	MEAN=	641.	STD DEV=	107.
351	94	157	182	222	265	466	596	643	677	756	MEAN=	372.	STD DEV=	230.
371	79	89	96	109	120	143	149	173	195	236	MEAN=	130.	STD DEV=	43.
393	67	78	81	87	90	96	102	111	120	136	MEAN=	94.	STD DEV=	19.
412	46	54	58	62	71	75	80	85	101	139	MEAN=	72.	STD DEV=	21.
433	34	43	45	47	64	80	90	92	96	103	MEAN=	67.	STD DEV=	24.
455	54	60	64	66	68	74	78	84	87	91	MEAN=	70.	STD DEV=	12.
476	47	54	55	60	70	77	81	90	99	196	MEAN=	75.	STD DEV=	30.
497	64	76	86	94	97	126	176	194	207	264	MEAN=	130.	STD DEV=	61.
518	0	0	0	2	57	64	100	169	185	214	MEAN=	71.	STD DEV=	75.
539	0	0	0	0	0	0	0	148	171	184	MEAN=	37.	STD DEV=	70.
560	0	0	0	0	0	0	0	0	0	2	MEAN=	0.	STD DEV=	0.
581	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV=	0.
602	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV=	0.
623	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV=	0.
644	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV=	0.

Table 9.5. The distributions of average man-days usage by monthly for the Shop 38 for the project.

		CUMULATIVE				PROBABILITIES								
		10%	20%	30%	40%	50%	60%	70%	80%	90%	100%			
SHOP	2	2273	2346	2419	2496	2544	2554	2646	2729	2774	2858	MEAN=	2540.	STD DEV= [illegible]
SHOP	3	166	171	173	179	180	182	185	186	196	201	MEAN=	180.	STD DEV= [illegible]
SHOP	5	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV= [illegible]
SHOP	6	1509	1549	1553	1583	1603	1630	1659	1697	1733	1765	MEAN=	1615.	STD DEV= [illegible]
SHOP	7	24	25	25	25	25	26	26	26	27	28	MEAN=	26.	STD DEV= [illegible]
SHOP	9	246	257	261	267	270	275	279	281	283	290	MEAN=	269.	STD DEV= [illegible]
SHOP	10	52	54	54	55	56	57	58	59	61	63	MEAN=	56.	STD DEV= [illegible]
SHOP	11	12199	12258	12273	12284	12298	12314	12349	12379	12422	12545	MEAN=	12313.	STD DEV= [illegible]
SHOP	12	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV= [illegible]
SHOP	13	142	151	154	159	160	165	168	171	180	189	MEAN=	161.	STD DEV= [illegible]
SHOP	14	4	4	4	4	5	5	5	5	5	5	MEAN=	5.	STD DEV= [illegible]
SHOP	16	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV= 0.
SHOP	17	8857	8933	9020	9087	9127	9206	9226	9249	9343	9405	MEAN=	9113.	STD DEV= [illegible]
SHOP	19	2641	2740	2771	2812	2883	2920	2981	3043	3139	3174	MEAN=	2884.	STD DEV= [illegible]
SHOP	20	51	55	56	58	59	60	61	62	64	67	MEAN=	59.	STD DEV= [illegible]
SHOP	22	159	166	169	173	179	182	189	196	201	205	MEAN=	179.	STD DEV= [illegible]
SHOP	23	239	242	243	246	246	248	248	249	251	254	MEAN=	246.	STD DEV= [illegible]
SHOP	24	472	485	525	541	563	600	604	629	646	661	MEAN=	562.	STD DEV= 63.
SHOP	26	12734	12879	12916	12931	12944	13017	13097	13108	13193	13269	MEAN=	12984.	STD DEV= 154.
SHOP	30	1495	1554	1606	1633	1653	1671	1681	1684	1710	1847	MEAN=	1635.	STD DEV= 79.
SHOP	31	14076	14283	14332	14355	14410	14477	14514	14539	14612	14739	MEAN=	14399.	STD DEV= [illegible]
SHOP	32	1097	1171	1179	1199	1215	1262	1286	1363	1381	1459	MEAN=	1245.	STD DEV= 103.
SHOP	33	331	337	344	350	351	353	359	362	369	376	MEAN=	351.	STD DEV= 14.
SHOP	34	2430	2540	2590	2664	2684	2757	2873	2914	3040	3168	MEAN=	2731.	STD DEV= 214.
SHOP	35	2228	2341	2445	2517	2532	2557	2607	2657	2738	2986	MEAN=	2526.	STD DEV= 189.
SHOP	36	2	2	2	2	2	3	3	3	3	3	MEAN=	2.	STD DEV= 0.
SHOP	37	2	2	2	2	3	3	3	3	3	3	MEAN=	3.	STD DEV= 0.
SHOP	38	26259	26464	26632	26718	26812	26949	27030	27223	27378	27485	MEAN=	26841.	STD DEV= 352.
SHOP	39	1830	2021	2079	2229	2296	2350	2398	2463	2560	2616	MEAN=	2250.	STD DEV= 247.
SHOP	40	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV= 0.
SHOP	41	6093	6184	6210	6295	6341	6364	6389	6403	6457	6491	MEAN=	6302.	STD DEV= [illegible]
SHOP	42	16	17	18	19	20	21	22	22	23	23	MEAN=	20.	STD DEV= 2.
SHOP	47	1	1	1	1	1	1	1	1	1	1	MEAN=	1.	STD DEV= 0.
SHOP	50	0	0	0	0	0	0	0	0	0	0	MEAN=	0.	STD DEV= 0.
SHOP	51	20687	20877	20944	21053	21114	21231	21348	21397	21701	21858	MEAN=	21155.	STD DEV= 334.
SHOP	56	41896	42138	42355	42697	42853	42967	43037	43287	43593	43951	MEAN=	42769.	STD DEV= [illegible]
SHOP	57	14167	14732	14910	15011	15118	15505	15643	15859	16081	16297	MEAN=	15241.	STD DEV= [illegible]
SHOP	62	1282	1320	1359	1431	1452	1493	1534	1563	1692	1709	MEAN=	1462.	STD DEV= [illegible]
SHOP	64	5101	5172	5292	5409	5538	5590	5664	5722	5874	6005	MEAN=	5490.	STD DEV= [illegible]
SHOP	67	12969	13114	13134	13215	13298	13372	13482	13532	13691	13750	MEAN=	13322.	STD DEV= 236.
SHOP	70	26	27	28	29	29	31	32	33	35	36	MEAN=	30.	STD DEV= 3.
SHOP	71	12170	12335	12398	12619	12659	12732	12776	12806	12951	13368	MEAN=	12611.	STD DEV= 289.
SHOP	72	16234	16673	16942	17063	17217	17446	17609	17778	18550	18742	MEAN=	17360.	STD DEV= 740.
SHOP	81	333	335	339	342	342	343	345	346	349	354	MEAN=	342.	STD DEV= 6.
SHOP	94	99	100	101	101	102	102	103	104	105	107	MEAN=	102.	STD DEV= 2.
SHOP	99	10560	10994	11163	11526	11672	11835	12206	12359	12710	13103	MEAN=	11694.	STD DEV= 750.
TOTAL		239970	241523	242013	242535	243432	243762	244227	244504	245222	247064	MEAN=	243094.	STD DEV= 1913.

Table 9.6. The distributions of total man-days from shops used for the completion of the project.

Stochastic simulations have been used to obtain unbiased distributions for important project statistics. The simulation model, by using flexible resource application rates in deciding the resource allocations among activities, by characterizing the stochastic man-hour requirements for production activities, and by characterizing the stochastic network structure of test activities, gave 'good' estimates for distributions for project schedule parameters. Having 'good' distributions for project parameters will improve the project planning and scheduling process in all levels (inter-project planning, project planning, project control) in the shipyard.

APPENDIX A1 Restriction on Intensity by a Progress Lag

According to the definition of transfer coefficient f_{ji} , at every intensity assignment, activity intensities must be assigned so that the progress lags from predecessor activities are not violated.

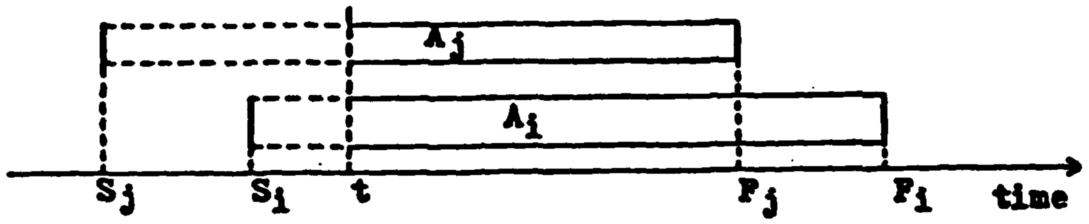


Figure A.1 Intermediate Product Transfer

Figure A.1 presents two overlapping activities, A_i and A_j . A_j is a predecessor of A_i . For the variable intensity case equation (5.5) must be satisfied for the flow transfers. Rewriting the equation (5.5) after substituting cumulative intensity terms (pcv_i) gives

$$pcv_i(t) \leq \begin{cases} Max\left\{0, pcv_j(t) - f_{ij}\right\} & \text{if } 0 \leq pcv_j(t) < 1.0 \\ 1.0 & \text{if } \quad pcv_j(t) = 1.0 \end{cases}$$

For the case $pcv_j(t)=1.0$, choosing intensity for A_i is independent from its predecessor A_j. For the case of $0 \leq pcv_j(t) < 1.0$ the following inequality must hold.

$$pcv_i(t) \leq Max\left\{0\ ,pcv_i(t) - f_{ji}\right\}$$

The algorithms do not consider assigning intensity to A_i before time t_1, where t_1 is such that $pcv_i(t_1)=f_{ji}$. We need to consider only the following case:

$$pcv_i(t) \le pcv_j(t) - f_{ji} \quad \text{for} \quad t \ge t_1$$

Suppose that the intensity $z_i(t)$ assigned to A_i will be maintained until $t+1$. Then the following inequality must hold:

$$pcv_i(t) + z_i(t) \le pcv_j(t) - f_{ji} + z_j(t)$$

Rearranging this gives

$$z_i(t) \ge pcv_j(t) - pcv_i(t) - f_{ji} + z_j(t) \tag{A1.1}$$

This is the actual restriction on intensity of A_i at time t. This assumes that the intensity of A_i will be changed at the beginning of each time period. We do not want to change the intensity assignments every period for only flow transfer reasons. Instead, we will assign a conservative intensity level to A_i such that it will satisfy the progress lag requirements for all periods assuming that the assigned intensity of A_i and current intensity of A_j will be kept constant until A_j finishes. Derivation of the conservative (more restricted) limit on intensities by flow transfers is as follows.

For the unit intensity case , when the following inequality holds, then inequalities (5.4) and (5.5) both hold.

$$F_i(t) - F_j(t) \ge f_{ji}\, d_i$$

Generalizing this inequality for the varying intensity case, assume that activity j will keep its intensity level at t until it finishes ($F_j(t)$), and also assume the intensity level decided for activity i will be kept constant until the activity finishes ($F_i(t)$). Then we require

$$F_i(t) - F_j(t) \ge f_{ji}\, \frac{1.}{z_i(t)}$$

Since,

$$F_i(t) = t + (1 - pcv_i(t)) \frac{1.}{z_i(t)}$$

by substituting,

$$t + (1 - pcv_i(t)) \frac{1.}{z_i(t)} - F_j(t) \geq f_{ji} \frac{1.}{z_i(t)}$$

Rearranging,

$$z_i(t) \leq z_i^* = \frac{(1.0 - pcv_i(t) - f_{ji})}{F_j(t) - t} \qquad \text{(A1.2)}$$

This must hold for all predecessors A_j of A_i.

The equation (A1.2) imposes a tighter restriction on intensity of A_i than the equation (A1.1) does, as can be seen from the following argument.

Since

$$F_j(t) = t + (1.0 - pcv_j(t)) \frac{1.0}{z_j(t)}$$

by substituting this expression into the equation (A1.2) and rearranging gives

$$z_i(t) \leq \frac{(1.0 - pcv_i(t) - f_{ji})\, z_j(t)}{1.0 - pcv_j(t)}$$

From (A1.1) and (A1.2), we have to show the following inequality holds

$$pcv_j(t) - pcv_i(t) - f_{ji} + z_j(t) \geq \frac{(1.0 - pcv_i(t) - f_{ji})\, z_j(t)}{1.0 - pcv_j(t)} \qquad \text{(A1.3)}$$

Rearranging,

$$(pcv_j(t) - pcv_i(t) - f_{ji})(1.0 - pcv_j(t)) \geq (pcv_j(t) - pcv_i(t) - f_{ji})\, z_j(t)$$

Rearranging,

$$(1.0 - pcv_j(t)) \geq z_j(t)$$

This inequality holds, because the intensity of A_j , ie, fraction of A_j which will be covered in period t, cannot be greater than the uncovered part of the activity A_j. This is true by definition. Then the equation (A1.3) holds.

Figure A.2 demonstrates the difference between the restrictions imposed by the equation (A1.2) and (A1.2).

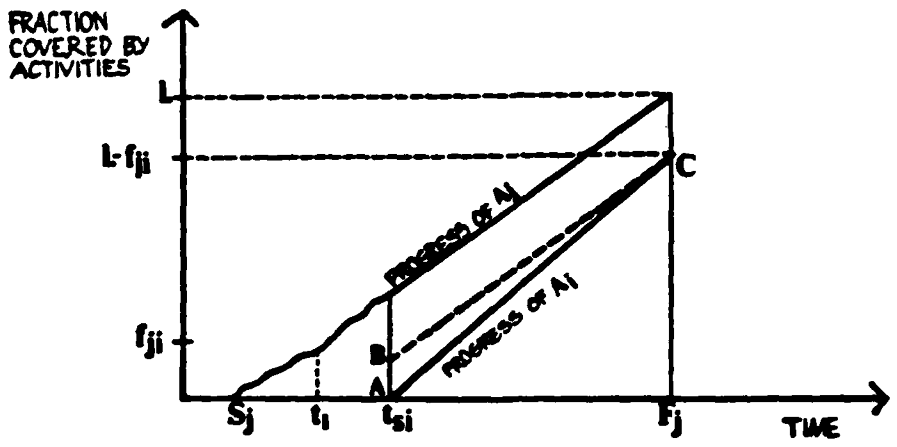


Figure A.2. Progresses of precedence related activities.

$\overline{AC}$ represents the progress curve if a_i^* is chosen as intensity level (Eq (A1.2)).$\overline{AB}$. $\overline{BC}$ represents the actual limitation imposed by the progress lag definition (Eq (A1.1)).

APPENDIX A2 Handling Storable Resources

Suppose that some of the resources are limited in their total (cumulative) consumption as well as their usage per period. Let

$R_l(t)$ Consumption limit of resource type l up to time t, ie, cumulative consumption of resource type l at time t cannot exceed this limit. It is an increasing function.

$$[\sum_{\tau=1}^{t} \sum_{i=1}^{N} a_{il}\ x_i(\tau) \leq R_l(t)]$$

Then, we need to modify the intensity assignment algorithms to handle this case. Define:

$AR_l(t)$ Available resource amount from resource type l at time t, ie, unused cumulative resource limit at time t.

$$= R_l(t) - \sum_{i=1}^{N} \sum_{\tau=1}^{t} a_{il}\ x_i(\tau)$$

Basically, the feasibility of cumulative resource consumption must be satisfied. This can be done by checking if there are available resources to consume for the activities in the next period.

The following modifications in intensity assignment algorithms are needed to handle this type of resource limitation:

(1) Initialize the cumulative available resources.

$AR_l = R_l(1)$ for $all\ l$

(2). In step 2(c), in deciding new $x_i(t)$, we need to consider the availability of cumulative resource consumptions. The equation for $y_i(t)$ will be changed to the following form.

$$y_i(t) = Min \begin{cases} \bar{z}_i & \textit{Intensity Upper Bound} \\ \min\limits_k \left\{ \dfrac{ar_k(t)}{a_{ki}} \right\} + z_i(t-1) & \textit{Resource Availiblity} \\ \min\limits_l \left\{ \dfrac{AR_l(t)}{a_{li}} \right\} + z_i(t-1) & \textit{Resource Consumption Availablity} \\ (1.0 - pcv_i(t-1)) \dfrac{1.0}{LF_i(t-1)-t} & \textit{Target Finish Time} \\ \min\limits_{j \epsilon \Gamma_i} \left\{ (1.0 - pcv_i(t-1) - f_{ji}) \dfrac{1.0}{F_j(t-1)-(t-1)} \right\} & \textit{Flow Transfer} \end{cases}$$

then update;

$AR_l(t) \leftarrow AR_l(t) - a_{li}(z_i(t) - z_i(t-1))$ for *all* l if $z_i(t) > 0$.

(3) In step 3(c-i), The resources for the rescheduled activities must be withdrawn.

$AR_l(t) = AR_l(t) + \sum_{\tau=1}^{t} a_{lj} \; z_i(\tau)$ for *all* l for *all* $j \epsilon IRESCHED$

(4) In step 4 , the same change shown for step 2(c).

(5) In step 5d, following update will be added

$$AR_l(\tau) \leftarrow AR_l(\tau-1) - a_{li} \; z_i(t), \qquad \tau = t, \ldots, t_n$$

and if $AR_l(\tau)$ becomes negative for any τ, then apply capacity change routine in order to recover the infeasibility.

(6) Change in limit of consumption will be treated as an event which is called a ' consumption limit change '. The event occurs when $R_k(t) > R_k(t-1)$ for any k. When there is such an event, the following updates are done. (Let $\Delta R_k(t) = R_k(t) - R_k(t-1)$.)

$AR_k(t) = AR_k(t-1) + \Delta R_k(t)$

then, go to intensity assignment.

These changes enable the algorithms to establish feasible schedules when there are resources that have limits on their total resource consumptions.

APPENDIX A3 Generating Arbitrary Project Networks Using Randomized Incidence Matrices

The first step of artificial network generation was the generation of an arbitrary incidence matrix that represents the precedence relations of project activities. The incidence matrix has N rows and N columns, where N is the number of activities in the project which is randomized between 30 and 50.

Let X_{ij} be the element of incidence matrix at ith row and jth column.

$$X_{ij} = \begin{cases} 1 & \textit{if } A_i \textit{ is followed by } A_j \\ 0 & \textit{Otherwise} \end{cases}$$

First, we set all elements of the incidence matrix to 1. (Later we will put some elements to 0 to eliminate the precedence relations that are not possible.) Then, the possible relations will be randomized in order to get the final form of the incidence matrix.

The project networks are acyclic; therefore, the part of the matrix that is under the diagonal must be set to zero.

$$X_{ij} = 0 \quad \text{if} \quad i \geq j$$

In project networks, it is unlikely to have precedence relations between the activities which are at the beginning of the network and the activities which are at the very end of the network. In order to avoid such a possibility, one can put a cut which is parallel to and above the diagonal of the matrix. The column where the cut starts determines the width of the network (M). The column M where the cut starts is randomized within [(N/2)-5, (N/2)+5]. Once M is decided, the cut will be defined as follows,

$$X_{ij} = 0 \qquad \text{if} \qquad M \le j-i$$

The resulting incidence matrix is shown in Figure A3.1. The elements within the region ABCD represent the possible precedence relations. The next step will be to randomly define zero elements within this region.

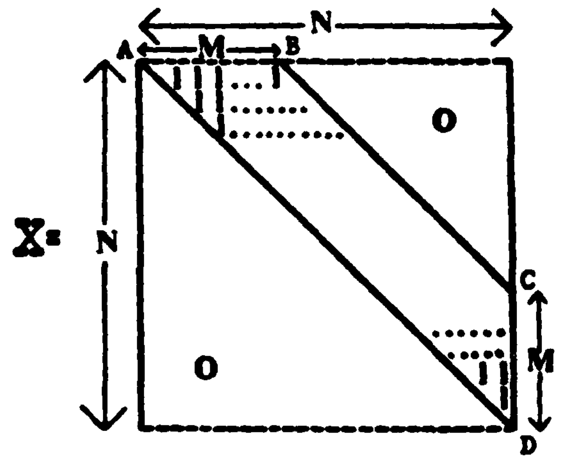


Figure A3.1. Project network incidence matrix

The number of followers (F) for each activity is randomized between 1.5 to 4.5. Note that most of the project networks that are used in the literature fall within this range. This is done as follows. The maximum possible number of followers of A_i is K_i , where

$$K_i = \sum_j X_{ij}$$

In order for A_i to have F followers on average, we choose the following probability level for each possible follower of A_i:

$$Pr_i = \frac{F}{K_i}$$

Then, a random draw is made for each possible follower of A_i(ie, for each entry in row i which lies in region ABCD). Let rn be a random number from a uniform distribution within [0,1]. Then,

$$X_{ij} = 0 \qquad \text{if} \qquad rn > Pr_i$$

This procedure is performed for all activities of the project in order to find the final form of the incidence matrix which will be taken as precedence relations of the arbitrary project networks. Note that, the expected number of followers F is not the same as the network complexity. The network complexities that are reported in Table 7.1 are calculated after incidence matrixes are finalized.

REFERENCES

Bitran, G. R. and Hax, A. C.

(1981) "Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables," *Management Science*, **27** (4), 431-441

Brand, J. D., Meyer, W. L. and Shaffer, L. R.

(1964) "The Resource Scheduling Method in Construction," Civil Engineering Studies Report, University of Illinios, Urbana

Boysen, J.

(1982) "Aggregate Project Model for Resource Allocation Within Multiproject Construction Systems," Unpublished PhD. Thesis, University of California-Berkeley

Burgess, A. R.,and Killebrew J. B.

(1962) "Variation in Activity Level on a Critical Arrow Diagram," *Journal of Industrial Engineering*, 13 (2), 76-83.

Clark, C. E

(1961) "The Greatest of Finite Set of Random Variables," *Operations Research*, **9** (2), 145-162

Clark, C. E.

(1962) "The PERT Model for the Distribution of an Activity Time," *Operations Research*, **10** (3), 405-406

Clingen, C. T.

(1964) "A Modification of Fulkerson's PERT Algorithm," *Operations Research*, 12 (4), 629-632

Cooper, D. F.

(1976) "Heuristics for Scheduling Resource Constrained Projects: An Experimental Investigation," *Management Science*, **22** (11), 1186-1194.

Davis, E. W.

(1973) "Project Scheduling under Resource Constraints - Historical Review and Categorization of Procedures," *AIIE Transactions*, 5 (4), 297-313.

Davis, E. W.,and Patterson, J. H.

(1975) "A Comparison of Heuristic and Optimal Solutions in Resource Constrained Project Scheduling," *Management Science*, 21 (8), 944-955.

DeWitte L.

(1964) "Manpower Leveling of PERT Networks," *Data Processing for Science/Engineering*, 2 (2), 29-37

Elmaghraby, S. E.

(1967) "On the Expected duration of PERT Type Networks," *Management Science*, 13 (5), 299-306

Elmaghraby, S. E.

(1977) "Activity Networks: Project Planning and Control by Network Models," John Wiley & Sons, New York

Fulkerson, D. R.,

(1961) "A Network Flow Computation for Project Cost Curves," *Management Science*, 7 (2), 167-168

Fulkerson, D. R.,

(1962) "Expected Critical Path Lengths in PERT Networks," *Operations Research*, 10 (5), 808-817

Gonguet, L.

(1969) "Comparison of Three Heuristic Procedures for Allocating Resources and Producing Schedule," in *Project Planning by Network Analysis*, Nort-Holland.

Grubbs, F. F.

(1962) "Attempts to Validate Certain PERT Statistics, or: Picking on PERT," *Operations Research*, 10 (6), 912-915

Hartley, H. D. and Woltham, A. W.

(1966) "Statistical Theory for PERT Critical Path Analysis," *Management Science*, 12 (10), 469-481

Hax, C. H. and Candea, D.

(1984) *Production and Inventory Management*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey

Herroelen, W.S.

(1972) "Resource Constrained Project Scheduling - the State of the Art," *Operational Research Quarterly*, 23 (3), 261-276.

Holloway, C. A.,Nelson, R. T. and Suraphangschai, V.

(1979) "Comparison of a Multi-Pass Heuristic Decomposition Procedure with other Resource Constrained Project Scheduling Procedures," *Management Science*, 25 , 862-872.

Johnson, T. J. R.

(1967) "An Algorithm for the Resource Constrained Project Scheduling Problem," Unpublished PhD. Thesis, Sloan Management School, M.I.T., Cambridge, Mass.

Kelley, J. E. Jr.

(1961) "Critical Path Planning and Scheduling: Mathematical Basis," *Operations Research*, 9 (3), 296-320

Kelley, J. E. Jr. and Walker M. R.

(1959) "Critical Path Planning and Scheduling," Proceedings of the Eastern Joint Computer Conference, Boston

Kurtulus, I. and E.W. Davis

(1982) "Multi-Project Scheduling: Categorization of Heuristic Rules Performance," *Management Science*, **28** (2), 161-172.

Levy, F. K., Thompson, G. L. and Wiest J. D.

(1962) "Multi-ship, Multi-shop, Workload Smoothing Program," *Naval Research Logistics Quarterly*, **9** (1), 37-44

Leachman, R.C.

(1982) "Production Planning for Multi-Resource Network Systems," *Naval Research Logistics Quarterly*, **29** (1), 47-54.

Leachman, R.C.

(1983) "Multiple Resource Leveling in Construction Systems Through Variation of Activity Intensities" *Naval Research Logistics Quarterly*, **30** (3), 187-195

Leachman, R. C. and Boysen, J.

(1984) "An Aggregate Model for Multi-Project Resource Allocations," in *Project Managemant: Methods and Studies*, B.V. Dean, editor, North Holland.

MacCrimmon, K. R. and Ryavec, C. A.

(1964) "An Analytical Study of PERT Assumptions," *Operations Research*, 12 (1), 16-36

Malcolm, D. G., Roseboom, J. H., Clark, C. E. and Fazar, W.

(1959) "Applications of a Technique for Research and Development Program Evaluation," *Operations Research*, 7 (5), 646- 669

Martin, C. C.

(1965) "Distribution of the Time Through a Directed Acyclic Network," *Operations Research*, 13 (1), 46-66

Meyer, W. L. and Shaffer, L. R.

(1963) "Extension of the Critical Path Method Through the Application of

Integer Programming," Report issued by the Department of Civil Engineering. University of Illinois, Urbana

Mize, J. H.

(1964) "A Heuristic Scheduling Model for Multi-Project Organizations," Unpublished Ph.D. Thesis, Purdue University.

Moder, J.J. and C.R. Phillips

(1970) *Project Managsment with CPM and PERT*, 2d ed., Van Nostrand Reinbold, New York.

Moeller, G. L.

(1972) "VERT" *Technical Papers: Twenty-Third Institute Conference and Convention*, American Institute of Industrial Engineers, Atlanta, Georgia

Moeller, G. L. and Digman, L. A.

(1981) "Operations Planning with VERT," *Operations Research*, **29** (4), 676-697

Moshman, J., Johnson, J. and Larsen, M.

(1963) "RAMPS-A Technique for Resource Allocation and Multi-Project Scheduling," *Joint Computer Proceedings*, **23**, Baltimore: Spartan

Mueller-Mehrback, H.

(1967) "Ein Verfaren Zur Planung des Optimalen Betriebsmitteleinsatzes Bei der Terminierung Van Grossprojekten," *AWM-Mitteilungen*, Frankfort

Pascoe, T. L.,

(1966) "Allocation of Resources in CPM," in *Revue Francaisem de Recherce Operutionnelle*, **38**

Patterson, J.H.

(1973) "Alternate Methods of Project Scheduling with Limited Resources," *Naval Research Logistics Quarterly*, **20** (4), 767-784.

Phillips, S. Jr. and Dessousky, M. I.

(1977) "Solving the Project Time/Cost Tradeoff Problem Using the Minimal Cut Concept," *Management Science*, **24** (4), 393-400

Pritsker, A. A. B. and Happ, W. W.

(1966) "GERT: Graphical Evaluation and Review Technique, Part I: Fundamentals," *Journal of Industrial Engineering* **17** (5), 267-274

Pritsker, A. A. B., Watters, L. S. and Wolfe, P. M.

(1969) "Milti-Project Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Science*, **16** (1), 39-108

Pritsker, A. A. B. and Whitehouse, G. E.,

(1966) "GERT: Graphical Evaluation and Review Technique, Part II: Probabilistic and Industrial Engineering Applications," *Journal of Industrial Engineering* **17** (6), 293-299

Robillard, P. and Trahan, M.

(1977) "The Completion Time of PERT Networks," *Operations Research*, **25** (1), 15-29

Shephard, R.W., R. Al-Ayat, and R.C. Leachman

(1977) "Shipbuilding Production Function: An Example of a Dynamic Production Function," in *Quantitative Wirtschaftsforschung Festschrift Volume (Wilhelm Krelle)*. H Albach et.al., Eds., JBCMorh. Tubingen, Germany

Shephard, R. W., and Mak, K

(1982) "Simulation of a Ship Overhaul Project Network" *Report ORC 82-3*, Operations Research Center, University of California, Berkeley.

Talbot, F. B.

(1982) "Resource-Constrained Project Scheduling with Time-Resource Tradeoffs : The Nonpreemptive Case," *Management Science*, **28** (10), 1197-

1210.

Talbot, F. B. and Patterson, J. H.,

(1978) "An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems," *Management Science*, **24** (11), 167-168

Thesen, A.

(1976) "Heuristic Scheduling of Activities Under Resource and Precedence Restrictions," *Management Science*, **23** (4), 412-422.

Van Slyke, R. M.

(1963) "Monte Carlo Methods and the PERT Problem," *Operations Research*, 11 (5), 839-860

Verbines, D. R.

(1963) "Optimum Scheduling of Limited Resources," *Chemical Engineering Progress*, **59** (3), 65-67

Weglarz, J.

(1976) "Time-Optimal Control of Resource Allocation in a Complex of Operations Framework," *IEEE Trans., Systems, Man and Cybernetics*, **6** 782-788.

Weglarz, J.

(1979) "Project Scheduling with Discrete and Continuous Resources," *IEEE Trans., Systems, Man and Cybernetics*, **9** 644-650.

Weglarz, J

(1981) "Project Scheduling With Continuously-Divisible, Doubly Constrained Resources," *Management Sciences*, **27** (9)

Whitehouse, G. E.

(1973) *System Analysis and Design Using Network Techniques*, Prentice-

Hall, Englewood Cliffs, N.J.

Wiest, Jerome D.

(1967) "A Heuristic Model for Scheduling Large Projects With Limited Resources," *Management Sciences*, **13** (6)